

```

EEEEEEEEEEEEEEEE XXX          XXX          CCCCCCCCCCCCC HHH          HHH          NNN          NNN          GGGGGGGGGGGGG
EEEEEEEEEEEEEEEE XXX          XXX          CCCCCCCCCCCCC HHH          HHH          NNN          NNN          GGGGGGGGGGGGG
EEEEEEEEEEEEEEEE XXX          XXX          CCCCCCCCCCCCC HHH          HHH          NNN          NNN          GGGGGGGGGGGGG
EEE          XXX          XXX          CCC          HHH          HHH          NNN          NNN          GGG
EEE          XXX          XXX          CCC          HHH          HHH          NNN          NNN          GGG
EEE          XXX          XXX          CCC          HHH          HHH          NNN          NNN          GGG
EEE          XXX          XXX          CCC          HHH          HHH          NNN          NNN          GGG
EEE          XXX          XXX          CCC          HHH          HHH          NNN          NNN          GGG
EEE          XXX          XXX          CCC          HHH          HHH          NNN          NNN          GGG
EEE          XXX          XXX          CCC          HHH          HHH          NNN          NNN          GGG
EEEEEEEEEEEEEEEE          XXX          XXX          CCC          HHH          HHH          NNN          NNN          GGG
EEEEEEEEEEEEEEEE          XXX          XXX          CCC          HHH          HHH          NNN          NNN          GGG
EEEEEEEEEEEEEEEE          XXX          XXX          CCC          HHH          HHH          NNN          NNN          GGG
EEEEEEEEEEEEEEEE          XXX          XXX          CCC          HHH          HHH          NNN          NNN          GGG
EEEE          XXX          XXX          CCC          HHH          HHH          NNN          NNN          GGG
EEE          XXX          XXX          CCC          HHH          HHH          NNN          NNN          GGG
EEE          XXX          XXX          CCC          HHH          HHH          NNN          NNN          GGG
EEE          XXX          XXX          CCC          HHH          HHH          NNN          NNN          GGG
EEE          XXX          XXX          CCC          HHH          HHH          NNN          NNN          GGG
EEEE          XXX          XXX          CCC          HHH          HHH          NNN          NNN          GGG
EEEEEEEEEEEEEEEE XXX          XXX          CCCCCCCCCCCCC HHH          HHH          NNN          NNN          GGGGGGGGGG
EEEEEEEEEEEEEEEE XXX          XXX          CCCCCCCCCCCCC HHH          HHH          NNN          NNN          GGGGGGGGGG
EEEEEEEEEEEEEEEE XXX          XXX          CCCCCCCCCCCCC HHH          HHH          NNN          NNN          GGGGGGGGGG

```

```

LL          IIIIII      SSSSSSSS
LL          IIIIII      SSSSSSSS
LL          II         SS
LL          II         SS
LL          II         SS
LL          II         SS
LL          II         SSSSSS
LL          II         SSSSSS
LL          II         SS
LL          II         SS
LL          II         SS
LL          II         SS
LLLLLLLLLLL IIIIIII    SSSSSSSS
LLLLLLLLLLL IIIIIII    SSSSSSSS

```

```
0001 0 MODULE  exch$init                                %TITLE 'INIT verb dispatch and misc routines'
0002 0
0003 0 IDENT = 'V04-000'
0004 0 ADDRESSING_MODE (EXTERNAL=LONG_RELATIVE, NONEXTERNAL=WORD_RELATIVE)
0005 0 ) =
0006 1 BEGIN
0007 1
0008 1 *****
0009 1 *
0010 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0011 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0012 1 * ALL RIGHTS RESERVED.
0013 1 *
0014 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0015 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0016 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0017 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0018 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0019 1 * TRANSFERRED.
0020 1 *
0021 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0022 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0023 1 * CORPORATION.
0024 1 *
0025 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0026 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0027 1 *
0028 1 *
0029 1 *****
0030 1
0031 1 ++
0032 1 FACILITY:      EXCHANGE - Foreign volume interchange facility
0033 1
0034 1 ABSTRACT:      Primary action routines for INIT verb
0035 1
0036 1 ENVIRONMENT:   VAX/VMS User mode
0037 1
0038 1 AUTHOR:        CW Hobbs          CREATION DATE: 04-Jan-1983
0039 1
0040 1 MODIFIED BY:
0041 1
0042 1          V03-002 CWH3002          CW Hobbs          12-Apr-1984
0043 1          Signal a specific error for an attempt to access a remote node
0044 1
0045 1
0046 1 --
0047 1
0048 1 ! Include files:
0049 1
0050 1 MACRO $module_name string = 'exch$init' %;          ! The require file needs to know our module name
0051 1 REQUIRE 'SRC$EXCREQ'                                ! Facility-wide require file
0052 1
```



```
54 0149 1 %SBTTL 'Module table of contents'
55 0150 1
56 0151 1 ! Module table of contents:
57 0152 1
58 0153 1 FORWARD ROUTINE
59 0154 1     init_dos11_init,           ! DOS-11 volume init processing
60 0155 1     init_foreign_close,    ! Close a foreign volume
61 0156 1     init_foreign_create,   ! Open a file to a foreign virtual volume
62 0157 1     init_foreign_open,     ! Open a file to a foreign device
63 0158 1     init_init : NOVALUE,    ! Setups
64 0159 1     exch$init_initialize,   ! Main action routine
65 0160 1     init_rt11_init,        ! RT-11 volume init processing
66 0161 1     init_zero_home_blocks ! Zero Files-11 home blocks
67 0162 1
68 0163 1
69 0164 1 ! EXCHANGE facility routines
70 0165 1
71 0166 1 EXTERNAL ROUTINE
72 0167 1     exch$cmd_cli_get_integer, ! Get the value of an integer qualifier
73 0168 1     exch$cmd_parse_filespec, ! Parse a file specification
74 0169 1     exch$io_dos11_rewind,    ! Rewind the sequential device
75 0170 1     exch$io_dos11_set_density, ! Set magtape density
76 0171 1     exch$io_dos11_write_tape_mark, ! Write a tape mark
77 0172 1     exch$io_rt11_write,     ! Write blocks to RT11 device
78 0173 1     exch$moun_vms_mount,     ! Perform VMS $mount service to mount volume
79 0174 1     exch$rt11_format_current_date : NOVALUE jsb_r1,
80 0175 1     exch$rtacp_verify_directory, ! Check for valid RT-11 directory
81 0176 1     exch$util_file_error,    ! Signal RMS error
82 0177 1     exch$util_namb_release : NOVALUE, ! Release name block
83 0178 1     exch$util_vm_allocate_zeroed, ! Allocate virtual memory
84 0179 1     exch$util_vm_release : NOVALUE, ! Release memory
85 0180 1     exch$util_vol_getdvi,    ! Get device information
86 0181 1     exch$util_volb_release : NOVALUE, ! Release volume block
87 0182 1     exch$util_volb_allocate ! Allocate volume block
88 0183 1
89 0184 1
90 0185 1 ! Equated symbols:
91 0186 1
92 0187 1 ! LITERAL
93 0188 1
94 0189 1
95 0190 1 ! Bound declarations:
96 0191 1
97 0192 1 ! BIND
98 0193 1
```

```
100 0194 1 GLOBAL ROUTINE init_dos11_init =
101 0195 2 BEGIN
102 0196 3 ++
103 0197 4
104 0198 5 FUNCTIONAL DESCRIPTION:
105 0199 6
106 0200 7     Perform dos11 volume specific init actions
107 0201 8
108 0202 9 INPUTS:
109 0203 10
110 0204 11     none
111 0205 12
112 0206 13 IMPLICIT INPUTS:
113 0207 14
114 0208 15     work area for INIT
115 0209 16
116 0210 17 OUTPUTS:
117 0211 18
118 0212 19     none
119 0213 20
120 0214 21 IMPLICIT OUTPUTS:
121 0215 22
122 0216 23     none
123 0217 24
124 0218 25 ROUTINE VALUE:
125 0219 26
126 0220 27     Success or worst error encountered.
127 0221 28
128 0222 29 SIDE EFFECTS:
129 0223 30
130 0224 31     dos11 tape will be initialized
131 0225 32
132 0226 33 --
133 0227 34 $dbgtrc_prefix ('init_dos11_init> ');
134 0228 35
135 0229 36 LOCAL
136 0230 37     dens,
137 0231 38     dosv : $ref_bblock,
138 0232 39     ent : $ref_bblock,
139 0233 40     status
140 0234 41 ;
141 0235 42
142 0236 43 BIND
143 0237 44     init = exch$a_gbl [exchg$a_init_work] : $ref_bblock, ! pointer to our work area
144 0238 45     volb = init [init$a_volb] : $ref_bblock ! pointer to exchange VOLB structure
145 0239 46 ;
146 0240 47
147 0241 48 $block_check (2, .init, init, 604);
148 0242 49 $block_check (2, .volb, volb, 605);
149 0243 50
150 0244 51 ! Make sure that we can do it
151 0245 52
152 0246 53 IF NOT .volb [volb$v_write]
153 0247 54 THEN
154 P 0248 55     $exch_signal_return ($warning_stat_copy (exch$ writelock), 2,
155 0249 56     .volb [volb$l_vol_ident_len], volb [volb$t_vol_ident]);
156 0250 57
```



```
157 0251 2 ! Allocate and initialize our volb extension if it does not exist
158 0252
159 0253 dosv = .volb [volb$a_vfmt_specific];
160 0254 IF .dosv EQL 0
161 0255 THEN
162 0256 BEGIN
163 0257 dosv = exch$util_vm_allocate_zeroed (exchblk$s_dos11); ! Get the memory
164 0258 volb [volb$a_vfmt_specific] = .dosv; ! Stash the address in the volb
165 0259 $block_init (.dosv, dos11); ! Set the type
166 0260 $queue_initialize (dosv [dos11$q_entry_header]); ! Init the directory cache queue
167 0261 END;
168 0262
169 0263 ! Rewind the magtape, then write two tape marks, then rewind the tape again
170 0264
171 0265 IF (status = exch$io_dos11_rewind (.volb))
172 0266 THEN
173 0267 IF (status = exch$io_dos11_write_tape_mark (.volb))
174 0268 THEN
175 0269 IF (status = exch$io_dos11_write_tape_mark (.volb))
176 0270 THEN
177 0271 status = exch$io_dos11_rewind (.volb);
178 0272
179 0273 ! If the /DENSITY qualifier is present, set the drive to the new density. Tape must be at BOT to change den
180 0274
181 0275 IF .status
182 0276 THEN
183 0277 IF cli$present (XASCII 'DENSITY')
184 0278 THEN
185 0279 status = exch$io_dos11_set_density (.volb);
186 0280
187 0281 ! If there is a cached "directory", release it
188 0282
189 0283 IF .dosv [dos11$a_entry_flink] NEQ 0
190 0284 THEN
191 0285 WHILE ((ent = $queue_remove_head (dosv [dos11$q_entry_header])) NEQ 0)
192 0286 DO
193 0287 exch$util_vm_release (dos11ent$k_length, .ent);
194 0288
195 0289 RETURN .status;
196 0290 1 END;
```

```
.TITLE EXCH$INIT INIT verb dispatch and misc routines
.IDENT \V04-000\
```

```
.PSECT EXCH$INIT_PLIT,NOWRT,2
```

```
00 59 54 49 53 4E 45 44 00000 P.AAB:
010E0007 00008 P.AAA:
00000000 0000C
```

```
.ASCII \DENSITY\<0>
.LONG 17694727
.ADDRESS P.AAB
```

```
.EXTRN EXCH$CMD_CLI_GET_INTEGER
.EXTRN EXCH$CMD_PARSE_FILESPEC
.EXTRN EXCH$IO_DOS11_REWIND
.EXTRN EXCH$IO_DOS11_SET_DENSITY
.EXTRN EXCH$IO_DOS11_WRITE_TAPE_MARK
.EXTRN EXCH$IO_RT11_WRITE
```

					.EXTRN	EXCH\$MOUN VMS MOUNT		
					.EXTRN	EXCH\$RT11 FORMAT CURRENT DATE		
					.EXTRN	EXCH\$RTACP VERIFY DIRECTORY		
					.EXTRN	EXCH\$UTIL_FILE_ERROR		
					.EXTRN	EXCH\$UTIL_NAMB_RELEASE		
					.EXTRN	EXCH\$UTIL_VM_ALLOCATE_ZEROED		
					.EXTRN	EXCH\$UTIL_VM_RELEASE		
					.EXTRN	EXCH\$UTIL_VOC_GETDVI		
					.EXTRN	EXCH\$UTIL_VOLB_RELEASE		
					.EXTRN	EXCH\$UTIL_VOLB_ALLOCATE		
					.EXTRN	EXCH\$A GBL, EXCH\$UTIL_BLOCK_CHECK		
					.EXTRN	EXCH\$ WRITELOCK		
					.EXTRN	CLISPRESENT		
					.PSECT	EXCH\$INIT_CODE, NOWRT, 2		
					.ENTRY	INIT DOS11_INIT, Save R2,R3,R4,R5,R6,R7		0194
					MOVAB	EXCH\$IO_DOS11_WRITE TAPE_MARK, R7		
					MOVAB	EXCH\$IO_DOS11_REWIND, R6		
					MOVAB	EXCH\$UTIL_BLOCK_CHECK, R5		
53	00000000G				ADDL3	#16, EXCH\$A GBL, R3		0237
54					ADDL3	#4, (R3), R4		0238
					MOVL	#2883833, R2		0241
					MOVZWL	#604, R1		
					MOVL	(R3), R0		
					JSB	EXCH\$UTIL_BLOCK_CHECK		
					MOVL	(R4), R3		0242
					MOVL	#68878579, R2		
					MOVZWL	#605, R1		
					MOVL	R3, R0		
					JSB	EXCH\$UTIL_BLOCK_CHECK		
22	48	A3			BBS	#5, 72(R3), 1\$		0246
		50	00000000G		MOVL	#EXCH\$ WRITELOCK, STATUS2		0249
		50			BICB2	#7, STATUS2		
		52			MOVL	STATUS2, TEMP		
					PUSHAB	105(R3)		
					PUSHL	101(R3)		
					PUSHL	#2		
					PUSHL	TEMP		
					CALLS	#4, LIB\$SIGNAL		
					MOVL	TEMP, R0		
					RET			
					MOVL	84(R3), DOSV		0253
					BNEQ	2\$		0254
					PUSHL	#54		0257
					CALLS	#1, EXCH\$UTIL_VM_ALLOCATE_ZEROED		
					MOVL	R0, DOSV		
					MOVL	DOSV, 84(R3)		0258
					MOVW	#54, 8(DOSV)		0259
					MNEGB	#3, 10(DOSV)		
					MOVAB	18(DOSV), R0		0260
					MOVL	R0, (R0)		
					MOVL	R0, 4(R0)		
					PUSHL	R3		0265
					CALLS	#1, EXCH\$IO_DOS11_REWIND		
					MOVL	R0, STATUS		
					BLBC	STATUS, 3\$		

00FC 00000
57 00000000G EF 9E 00002
56 00000000G EF 9E 00009
55 00000000G EF 9E 00010
53 00000000G EF 10 C1 00017
54 63 04 C1 0001F
52 002C00F9 8F D0 00023
51 025C 8F 3C 0002A
50 63 D0 0002F
65 16 00032
53 64 D0 00034
52 041B00F3 8F D0 00037
51 025D 8F 3C 0003E
50 53 D0 00043
65 16 00046
22 48 A3 05 E0 00048
50 00000000G 8F D0 0004D
50 07 8A 00054
52 50 D0 00057
69 A3 9F 0005A
65 A3 DD 0005D
02 DD 00060
52 DD 00062
00000000G 00 04 FB 00064
50 52 D0 0006B
52 04 0006E
54 A3 D0 0006F 1\$:
23 12 00073
36 DD 00075
00000000G EF 01 FB 00077
52 50 D0 0007E
54 A3 52 D0 00081
08 A2 36 B0 00085
0A A2 03 8E 00089
50 12 A2 9E 0008D
60 50 D0 00091
04 A0 50 D0 00094
53 DD 00098 2\$:
66 01 FB 0009A
54 50 D0 0009D
3B 54 E9 000A0

EXCH\$INIT
V04-000

INIT verb dispatch and misc routines
init_dos11_init

K 6
16-Sep-1984 00:59:01
14-Sep-1984 12:29:05

VAX-11 Bliss-32 V4.0-742
[EXCHNG.SRC]EXCINIT.B32;1

Page 6
(3)

		53	DD	000A3	PUSHL	R3		0267
67		01	FB	000A5	CALLS	#1, EXCH\$IO_DOS11_WRITE_TAPE_MARK		
54		50	D0	000A8	MOVL	R0, STATUS		
30		54	E9	000AB	BLBC	STATUS, 3\$		
		53	DD	000AE	PUSHL	R3		0269
67		01	FB	000B0	CALLS	#1, EXCH\$IO_DOS11_WRITE_TAPE_MARK		
54		50	D0	000B3	MOVL	R0, STATUS		
25		54	E9	000B6	BLBC	STATUS, 3\$		
		53	DD	000B9	PUSHL	R3		0271
66		01	FB	000BB	CALLS	#1, EXCH\$IO_DOS11_REWIND		
54		50	D0	000BE	MOVL	R0, STATUS		
1A		54	E9	000C1	BLBC	STATUS, 3\$		0275
	00000000G	00	CF	9F 000C4	PUSHAB	P.AAA		0277
		01	FB	000C8	CALLS	#1, CLISPRESNT		
		50	E9	000CF	BLBC	R0, 3\$		
	00000000G	EF	53	DD	000D2	PUSHL	R3	0279
		54	01	FB	000D4	CALLS	#1, EXCH\$IO_DOS11_SET_DENSITY	
			50	D0	000DB	MOVL	R0, STATUS	
		12	A2	D5 000DE 3\$:	TSTL	18(DOSV)		0283
			1C	13 000E1	BEQL	7\$		
	50	12	B2	0F 000E3 4\$:	REMQUE	@18(DOSV), _T_		0285
			04	1C 000E7	BVC	5\$		
			53	D4 000E9	CLRL	ENT		
			03	11 000EB	BRB	6\$		
	53		50	D0 000ED 5\$:	MOVL	_T_, ENT		
			0D	13 000F0 6\$:	BEQL	7\$		
			53	DD 000F2	PUSHL	ENT		0287
			1C	DD 000F4	PUSHL	#28		
	00000000G	EF	02	FB 000F6	CALLS	#2, EXCH\$UTIL_VM_RELEASE		
			E4	11 000FD	BRB	4\$		
		50	54	D0 000FF 7\$:	MOVL	STATUS, R0		0289
			04	00102	RET			0290

; Routine Size: 259 bytes, Routine Base: EXCH\$INIT_CODE + 0000


```
198 0291 1 GLOBAL ROUTINE init_foreign_close = XSBTTL 'init_foreign_close'
199 0292 BEGIN
200 0293 ++
201 0294
202 0295 FUNCTIONAL DESCRIPTION:
203 0296
204 0297     Close a temporarily opened foreign device.
205 0298
206 0299 INPUTS:
207 0300
208 0301     none
209 0302
210 0303 IMPLICIT INPUTS:
211 0304
212 0305     INIT verb work area
213 0306
214 0307 OUTPUTS:
215 0308
216 0309     none
217 0310
218 0311 IMPLICIT OUTPUTS:
219 0312
220 0313     work area
221 0314
222 0315 ROUTINE VALUE:
223 0316
224 0317     Success or worst error encountered.
225 0318
226 0319 SIDE EFFECTS:
227 0320
228 0321     A file is no longer open on the volb
229 0322 --
230 0323
231 0324 $dbgtrc_prefix ('init_foreign_close> ');
232 0325
233 0326 LOCAL
234 0327     status
235 0328 ;
236 0329
237 0330 BIND
238 0331     init = exch$a_gbl [excg$a_init_work] : $ref_bblock, ! pointer to our work area
239 0332     volb = .init [init$a_volb] : $bblock, ! Pointer to exchange VOLB structure
240 0333     fab = .volb [volb$a_fab] : $bblock ! File Access Block for the volume
241 0334 ;
242 0335
243 0336 $block_check (2, volb, volb, 575);
244 0337
245 0338 ! Close the open RMS link to the volume
246 0339 !
247 0340 IF NOT (status = $close (fab = fab))
248 0341 THEN
249 0342     RETURN exch$util_file_error (exch$_closeforeign, .status, fab, .fab [fab$l_stv]);
250 0343
251 0344 RETURN .status;
252 0345 1 END;
```

EXCH\$INIT
V04-000

INIT verb dispatch and misc routines
init_foreign_close

M 6
16-Sep-1984 00:59:01
14-Sep-1984 12:29:05

VAX-11 Bliss-32 V4 ^ 742
[EXCHNG.SRC]EXCI..1.B32;1

Page 8
(4)

```

                                000C 00000
50 C0000000G EF              10 C1 00002
                                60 D0 0000A
                                A0 D0 0000D
                                53 10 A0 D0 00011
                                52 041B00F3 8F D0 00015
                                51 023F 8F 3C 0001C
                                00000000G EF 16 00021
                                53 DD 00027
00000000G 00              01 FB 00029
                                52 D0 00030
                                13 52 E8 00033
                                0C A3 DD 00036
                                0C BB 00039
                                00000000G 8F DD 0003B
00000000G EF              04 FB 00041
                                04 04 00048
                                50 52 D0 00049 1$:
                                04 0004C
```

.EXTRN SYSSCLOSE, EXCH\$_CLOSEFOREIGN

```

.ENTRY INIT_FOREIGN_CLOSE, Save R2,R3
ADDL3 #16, EXCH$A_GBL, R0
MOVL (R0), R0
MOVL 4(R0), R0
MOVL 16(R0), R3
MOVL #68878579, R2
MOVZWL #575, R1
JSB EXCH$UTIL_BLOCK_CHECK
PUSHL R3
CALLS #1, SYSSCLOSE
MOVL R0, STATUS
BLBS STATUS, 1$
PUSHL 12(R3)
PUSHR #^M<R2,R3>
PUSHL #EXCH$_CLOSEFOREIGN
CALLS #4, EXCH$UTIL_FILE_ERROR
RET
MOVL STATUS, R0
RET
```

```

: 0291
: 0331
: 0332
: 0333
: 0336
: 0340
: 0342
: 0344
: 0345
```

; Routine Size: 77 bytes, Routine Base: EXCH\$INIT_CODE + 0103

```
254 0346 1 GLOBAL ROUTINE init_foreign_create = %SBTTL 'init_foreign_create'
255 0347 BEGIN
256 0348 ++
257 0349
258 0350 FUNCTIONAL DESCRIPTION:
259 0351
260 0352 Create a foreign virtual volume with RMS so that we may initialize it.
261 0353
262 0354 INPUTS:
263 0355
264 0356 none
265 0357
266 0358 IMPLICIT INPUTS:
267 0359
268 0360 namb - name block describing the device
269 0361
270 0362 OUTPUTS:
271 0363
272 0364 none
273 0365
274 0366 IMPLICIT OUTPUTS:
275 0367
276 0368 volb - volume block which will describe the mounted volume
277 0369
278 0370 ROUTINE VALUE:
279 0371
280 0372 Success or worst error encountered.
281 0373
282 0374 SIDE EFFECTS:
283 0375
284 0376 lots
285 0377
286 0378 --
287 0379 $dbgtrc_prefix ('init_foreign_create> ');
288 0380
289 0381 LOCAL
290 0382 len,
291 0383 snum,
292 0384 start,
293 0385 status
294 0386 ;
295 0387
296 0388 BIND
297 0389 init = exch$a_gbl [excg$a_init_work] : $ref_bblock, | pointer to our work area
298 0390 fildesc = init [init$a_device] : $bblock, | file name
299 0391 namb = .init [init$a_namb] : $bblock, | Pointer to exchange NAMB structure
300 0392 volb = .init [init$a_volb] : $bblock, | Pointer to exchange VOLB structure
301 0393 fab = .volb [volb$a_fab] : $bblock, | File Access Block for the volume
302 0394 rab = .volb [volb$a_rab] : $bblock, | Record Access Block for the volume
303 0395 nam = .volb [volb$a_nam] : $bblock, | RMS name block for the volume
304 0396 dev_desc = namb [namb$a_device] : $desc_block | Pointer to the device name
305 0397 ;
306 0398
307 0399 $trace_print_lit ('entry');
308 0400 $block_check (2, .init, init, 630);
309 0401 $block_check (2, namb, namb, 631);
310 0402 $block_check (2, volb, volb, 632);
```



```
0403 2  ! Copy the input name to the volb for the signal
```

```
0404 2  !
0405 2  !
0406 2  len = MINU (volb$vol_ident, .fildesc [dsc$a_length]);
0407 2  CHSMOVE (.len, .fildesc [dsc$a_pointer], volb [volb$vol_ident]);
0408 2  volb [volb$vol_ident_len] = .len;
```

```
0409 2  ! Determine the number of device blocks
```

```
0410 2  !
0411 2  len = (BEGIN
0412 2  LOCAL
0413 2  bmax;
0414 2  bmax = MINU (65535, .init [init$L_q_allocation]);
0415 2  IF .bmax EQL 0
0416 2  THEN
0417 2  bmax = 494; ! Default to single density diskette
0418 2  IF .init [init$L_q_allocation] GTRU .bmax
0419 2  THEN
0420 2  $exch_signal (exch$rt11_toomanyblk, 1, .bmax);
0421 2  .bmax
0422 2  END);
```

```
0423 2  ! Determine the number of directory segments, so that we can put a floor on the size of the file
```

```
0424 2  !
0425 2  snum = (SELECTONE true OF
0426 2  SET
0427 2  [.init [init$L_q_segments] NEQ 0] : .init [init$L_q_segments];
0428 2  [.len LEQU 512] : 1;
0429 2  [.len LEQU 2048] : 4;
0430 2  [.len LEQU 12288] : 16;
0431 2  [OTHERWISE] : 31;
0432 2  TES);
```

```
0433 2  ! Apply the floor and determine the number of blocks
```

```
0434 2  !
0435 2  start = rt11$root_block + (2 * .snum);
0436 2  len = MAXU (.start+32, .len); ! Make it at least 32 blocks for files
0437 2  !
0438 2  volb [volb$devmaxblock] = .len; ! We need to save it here too
0439 2  !
0440 2  volb [volb$volmaxblock] = .len; ! We need to save it here too
```

```
0441 2  ! Init the RMS blocks for the volume
```

```
0442 2  !
0443 2  $fab_init (
0444 2  FAB = fab, ! Volume FAB
0445 2  ALQ = .len, ! Allocation quantity
0446 2  FAC = (BIO,GET,PUT), ! Block I/O, read and write
0447 2  FNA = .fildesc [dsc$a_pointer], ! Set name addr
0448 2  FNS = .fildesc [dsc$a_length], ! Set name size
0449 2  DNA = UPLIT BYTE ('VIRTUAL.DSK'), ! Default name address
0450 2  DNS = 11, ! Default name size
0451 2  MRS = 512, ! Records size
0452 2  RAT = CR, ! Carriage return
0453 2  RFM = FIX, ! Fixed length records
0454 2  NAM = nam); ! Name block
0455 2  !
0456 2  $rab_init (
0457 2  RAB = rab, ! Volume RAB
0458 2  ROP = BIO, ! Block I/O
```

```
368 0460      FAB = fab);
369 0461      $nam_init (
370 0462          NAM = nam,
371 0463          RSA = .volb [volb$a_rsbuf],
372 0464          RSS = nam$c_maxrss,
373 0465          ESA = .volb [volb$a_esbuf],
374 0466          ESS = nam$c_maxrss);
375 0467
376 0468      ! Create and connect to the volume
377 0469
378 0470      IF NOT (status = $create (fab = fab))
379 0471      THEN
380 0472          RETURN exch$util_file_error (exch$createvirt, .status, fab, .fab [fab$l_stv]);
381 0473
382 0474      ! Now put as much of the result name into the volb as we can
383 0475
384 0476      len = MINU (volb$s_vol_ident, nam [nam$b_rsl]);
385 0477      CHSMOVE (.len, nam [nam$l_rsa], volb [volb$t_vol_ident]);
386 0478      volb [volb$l_vol_ident_len] = .len;
387 0479
388 0480      volb [volb$w_channel] = .fab [fab$l_stv];      ! Save the channel number (NFS ==> user mode channel)
389 0481
390 0482      IF NOT (status = $connect (rab = rab))
391 0483      THEN
392 0484          RETURN exch$util_file_error (exch$createvirt, .status, fab, .rab [rab$l_stv]);
393 0485
394 0486      ! Set the volume format and other bits and pieces
395 0487
396 0488      volb [volb$b_vol_format] = volb$k_vfmt_rt11;
397 0489      volb [volb$v_write] = true;
398 0490      volb [volb$v_virtual] = true;
399 0491
400 0492      ! Write the last block to set the eof block correctly
401 0493
402 0494      IF NOT (status = exch$io_rt11_write (volb, .volb [volb$l_volmaxblock]-1, 1, exch$io_rt11_write))
403 0495      THEN
404 0496          RETURN .status;
405 0497
406 0498      RETURN true;
407 0499      END;
```

```
                                .PSECT EXCH$INIT_PLIT,NOWRT,2
4B 53 44 2E 4C 41 55 54 52 49 56 00010 P.AAC: .ASCII \VIRTUAL.DSK\
                                .EXTRN EXCH$ RT11_TOOMANYBLK
                                .EXTRN SYSS$CREATE, EXCH$_CREATEVIRT
                                .EXTRN SYSS$CONNECT
                                .PSECT EXCH$INIT_CODE,NOWRT,2
                                OFFC 00000
                                .ENTRY INIT FOREIGN_CREATE, Save R2,R3,R4,R5,R6,- : 0346
                                R7,R8,R9,R10,R11
                                ADDL3 #16, EXCH$a_GBL, R0 : 0389
                                MOVL (R0), R8 : 0390
```

		OC	A8	9F	0000D	PUSHAB	12(R8)		
	53		68	D0	00010	MOVL	(R8), R3	0391	
	57	04	A8	D0	00013	MOVL	4(R8), R7	0392	
	56	10	A7	D0	00017	MOVL	16(R7), R6	0393	
	5A	14	A7	D0	0001B	MOVL	20(R7), R10	0394	
	59	18	A7	D0	0001F	MOVL	24(R7), R9	0395	
	52	002C00F9	8F	D0	00023	MOVL	#2883833, R2	0400	
	51	0276	8F	3C	0002A	MOVZWL	#630, R1		
	50		58	D0	0002F	MOVL	R8, R0		
		00000000G	EF	16	00032	JSB	EXCH\$UTIL_BLOCK_CHECK		
	52	010A00F7	8F	D0	00038	MOVL	#17432823, R2	0401	
	51	0277	8F	3C	0003F	MOVZWL	#631, R1		
	50		53	D0	00044	MOVL	R3, R0		
		00000000G	EF	16	00047	JSB	EXCH\$UTIL_BLOCK_CHECK		
	52	041B00F3	8F	D0	0004D	MOVL	#68878579, R2	0402	
	51	0278	8F	3C	00054	MOVZWL	#632, R1		
	50		57	D0	00059	MOVL	R7, R0		
		00000000G	EF	16	0005C	JSB	EXCH\$UTIL_BLOCK_CHECK		
	50	00	BE	3C	00062	MOVZWL	20(SP), R0	0406	
0080	8F		50	B1	00066	CMPL	R0, #128		
			04	1B	0006B	BLEQU	1\$		
	50	80	8F	9A	0006D	MOVZBL	#128, R0		
	58		50	D0	00071	MOVL	R0, LEN		
7E	6E		04	C1	00074	ADDL3	#4, (SP), -(SP)	0407	
			9E	DD	00078	PUSHL	2(SP)+		
69	A7		5B	28	0007A	MOVCL	LEN, 2(SP)+ 105(R7)		
	65		5B	D0	0007F	MOVL	LEN, 101(R7)	0408	
		1C	A8	D0	00083	MOVL	28(R8), R0	0415	
	0000FFFF		50	D1	00087	CMPL	R0, #65535		
			05	1B	0008E	BLEQU	2\$		
	50	FFFF	8F	3C	00090	MOVZWL	#65535, R0		
	52		50	D0	00095	MOVL	R0, BMAX		
			05	12	00098	BNEQ	3\$	0416	
	52	01EE	8F	3C	0009A	MOVZWL	#494, BMAX	0418	
	52	1C	A8	D1	0009F	CMPL	28(R8), BMAX	0419	
			11	1B	000A3	BLEQU	4\$		
			52	DD	000A5	PUSHL	BMAX	0421	
			01	DD	000A7	PUSHL	#1		
		00000000G	8F	DD	000A9	PUSHL	#EXCH\$ RT11 TOOMANYBLK		
	00000000G	00	03	FB	000AF	CALLS	#3, LIB\$SIGNAL		
	5B		52	D0	000B6	MOVL	BMAX, LEN	0422	
		24	A8	D5	000B9	TSTL	36(R8)	0429	
			06	13	000BC	BEQL	5\$		
	50	24	A8	D0	000BE	MOVL	36(R8), SNUM		
			2D	11	000C2	BRB	9\$		
	00000200		5B	D1	000C4	CMPL	LEN, #512	0430	
			05	1A	000CB	BGTRU	6\$		
	50		01	D0	000CD	MOVL	#1, SNUM		
			1F	11	000D0	BRB	9\$		
	00000800		5B	D1	000D2	CMPL	LEN, #2048	0431	
			05	1A	000D9	BGTRU	7\$		
	50		04	D0	000DB	MOVL	#4, SNUM		
			11	11	000DE	BRB	9\$		
	00003000		5B	D1	000E0	CMPL	LEN, #12288	0432	
			05	1A	000E7	BGTRU	8\$		
	50		10	D0	000E9	MOVL	#16, SNUM		
			03	11	000EC	BRB	9\$		

0050	8F	00	50	1F	D0	000EE	8\$:	MOVL	#31, SNUM	0433
			50	02	C4	000F1	9\$:	MULL2	#2, START	0438
			50	26	C0	000F4		ADDL2	#38, R0	0439
			5B	50	D1	000F7		CMPL	R0, LEN	
				03	1E	000FA		BGEQU	10\$,	
			50	5B	D0	000FC		MOVL	LEN, R0	
			5B	50	D0	000FF	10\$:	MOVL	R0, LEN	
		40	A7	5B	D0	00102		MOVL	LEN, 64(R7)	0440
		44	A7	5B	D0	00106		MOVL	LEN, 68(R7)	0441
			6E	00	2C	0010A		MOVC5	#0, (SP), #0, #80, (R6)	0456
				66		00111				
			66	8F	B0	00112		MOVW	#20483, (R6)	
		10	A6	5B	D0	00117		MOVL	LEN, 16(R6)	
		16	A6	23	90	0011B		MOVB	#35, 22(R6)	
		1E	A6	8F	B0	0011F		MOVW	#258, 30(R6)	
		28	A6	59	D0	00125		MOVL	R9, 40(R6)	
			6E	04	C1	00129		ADDL3	#4, (SP), R0	
		2C	A6	60	D0	0012D		MOVL	(R0), 44(R6)	
		30	A6	CF	9E	00131		MOVAB	P.AAC, 48(R6)	
		34	A6	BE	90	00137		MOVB	#0(SP), 52(R6)	
		35	A6	0B	90	0013C		MOVB	#11, 53(R6)	
		36	A6	8F	B0	00140		MOVW	#512, 54(R6)	
0044	8F	00	6E	00	2C	00146		MOVC5	#0, (SP), #0, #68, (R10)	0460
				6A		0014D				
			6A	8F	B0	0014E		MOVW	#17409, (R10)	
		04	AA	8F	3C	00153		MOVZWL	#2048, 4(R10)	
		3C	AA	56	D0	00159		MOVL	R6, 60(R10)	
0060	8F	00	6E	00	2C	0015D		MOVC5	#0, (SP), #0, #96, (R9)	0466
				69		00164				
			69	8F	B0	00165		MOVW	#24578, (R9)	
		02	A9	01	8E	0016A		MNEGB	#1, 2(R9)	
		04	A9	A7	D0	0016E		MOVL	32(R7), 4(R9)	
		0A	A9	01	8E	00173		MNEGB	#1, 10(R9)	
		0C	A9	A7	D0	00177		MOVL	28(R7), 12(R9)	
				56	DD	0017C		PUSHL	R6	0470
		00000000G	00	01	FB	0017E		CALLS	#1, SYS\$CREATE	
			58	50	D0	00185		MOVL	R0, STATUS	
			05	58	E8	00188		BLBS	STATUS, 11\$	
				A6	DD	0018B		PUSHL	12(R6)	0472
				32	11	0018E		BRB	13\$	
			50	A9	9A	00190	11\$:	MOVZBL	3(R9), R0	0476
		80	8F	50	91	00194		CMPB	R0, #128	
				04	1B	00198		BLEQU	12\$	
			50	8F	9A	0019A		MOVZBL	#128, R0	
			5B	50	D0	0019E	12\$:	MOVL	R0, LEN	
69	A7		04	5B	28	001A1		MOVC3	LEN, 24(R9), 105(R7)	0477
			65	5B	D0	001A7		MOVL	LEN, 101(R7)	0478
			4A	A6	B0	001AB		MOVW	12(R6), 74(R7)	0480
				5A	DD	001B0		PUSHL	R10	0482
		00000000G	00	01	FB	001B2		CALLS	#1, SYS\$CONNECT	
			58	50	D0	001B9		MOVL	R0, STATUS	
			15	58	E8	001BC		BLBS	STATUS, 14\$	
				AA	DD	001BF		PUSHL	12(R10)	0484
				56	DD	001C2	13\$:	PUSHL	R6	
				58	DD	001C4		PUSHL	STATUS	
		00000000G	EF	8F	DD	001C6		PUSHL	#EXCH\$ CREATEVIRT	
				04	FB	001CC		CALLS	#4, EXCH\$UTIL_FILE_ERROR	

EXCH\$INIT
V04-000

INIT verb dispatch and misc routines
init_foreign_create

F 7
16-Sep-1984 00:59:01
14-Sep-1984 12:29:05

VAX-11 Bliss-32 V4.0-742
[EXCHNG.SRC]EXCINIT.B32;1

Page 14
(5)

58	A7	03	04	001D3	RET		
48	A7	30	90	001D4	148: MOVB	#3, 88(R7)	0488
		EF	88	001D8	BISB2	#48, 72(R7)	0490
	00000000G	01	9F	001DC	PUSHAB	EXCH\$IO_RT11_WRITE	0494
7E	44	01	DD	001E2	PUSHL	#1	
		57	C3	001E4	SUBL3	#1, 68(R7), -(SP)	
	00000000G	04	DD	001E9	PUSHL	R7	
		50	FB	001EB	CALLS	#4, EXCH\$IO_RT11_WRITE	
		58	D0	001F2	MOVL	R0, STATUS	
		58	E8	001F5	BLBS	STATUS, 15\$	
		58	D0	001F8	MOVL	STATUS, R0	0496
		04	04	001FB	RET		
	50	01	D0	001FC	15\$: MOVL	#1, R0	0498
		04	04	001FF	RET		0499

; Routine Size: 512 bytes, Routine Base: EXCH\$INIT_CODE + 0150

EXC
V04

```
409 0500 1 GLOBAL ROUTINE init_foreign_open = %SBTTL 'init_foreign_open'
410 0501 2 BEGIN
411 0502 3 ++
412 0503 4
413 0504 5 FUNCTIONAL DESCRIPTION:
414 0505 6
415 0506 7     Open a foreign device with RMS so that we may initialize it.
416 0507 8
417 0508 9 INPUTS:
418 0509 10
419 0510 11     none
420 0511 12
421 0512 13 IMPLICIT INPUTS:
422 0513 14
423 0514 15     namb - name block describing the device
424 0515 16
425 0516 17 OUTPUTS:
426 0517 18
427 0518 19     none
428 0519 20
429 0520 21 IMPLICIT OUTPUTS:
430 0521 22
431 0522 23     volb - volume block which will describe the mounted volume
432 0523 24
433 0524 25 ROUTINE VALUE:
434 0525 26
435 0526 27     Success or worst error encountered.
436 0527 28
437 0528 29 SIDE EFFECTS:
438 0529 30
439 0530 31     lots
440 0531 32
441 0532 33 --
442 0533 34 $dbgtrc_prefix ('init_foreign_open> ');
443 0534 35
444 0535 36 LOCAL
445 0536 37     status
446 0537 38 ;
447 0538 39
448 0539 40 BIND
449 0540 41     init = exch$a_gbl [excg$a_init work] : $ref_block, ! pointer to our work area
450 0541 42     namb = .init [init$a_namb] : $bblock, ! Pointer to exchange NAMB structure
451 0542 43     volb = .init [init$a_volb] : $bblock, ! Pointer to exchange VOLB structure
452 0543 44     fab = .volb [volb$a_fab] : $bblock, ! File Access Block for the volume
453 0544 45     rab = .volb [volb$a_rab] : $bblock, ! Record Access Block for the volume
454 0545 46     nam = .volb [volb$a_nam] : $bblock, ! RMS name block for the volume
455 0546 47     dev_desc = namb [namb$a_device] : $desc_block ! Pointer to the device name
456 0547 48 ;
457 0548 49
458 0549 50 $block_check (2, .init, init, 571);
459 0550 51 $block_check (2, namb, namb, 572);
460 0551 52 $block_check (2, volb, volb, 573);
461 0552 53
462 0553 54 ! Get the device information
463 0554 55
464 0555 56 IF NOT (status = exch$util_vol_getdvi (dev_desc, volb))
465 0556 57 THEN
```



```
466 0557 BEGIN
467 0558 $exch_signal (exch$_accessfail, 1, dev_desc, .status);
468 0559 RETURN .status;
469 0560 END;
470 0561
471 0562 ! Look at the device characteristics and make some decisions
472 0563
473 0564 BEGIN ! scope 'devbits'
474 0565 BIND
475 0566 devbits = volb [volb$_devchar] : $bblock;
476 0567 REGISTER
477 0568 must_have, cannot_have; ! masks for device tests
478 0569
479 0570 ! We need to make sure that the thing is at least similar to a disk or tape. First define masks for all
480 0571 ! required and all prohibited device characteristics
481 0572
482 0573 IF .devbits [dev$_v_rnd]
483 0574 THEN
484 0575 BEGIN ! bits for "disks"
485 0576 must_have = (dev$_m_rnd OR dev$_m_fod OR dev$_m_shr OR dev$_m_avl OR dev$_m_idv OR dev$_m_odv OR dev$_m_dir
486 0577 cannot_have = (dev$_m_rec OR dev$_m_ccl OR dev$_m_trm OR dev$_m_sdi OR dev$_m_sgd OR dev$_m_spl OR dev$_m_o
487 0578 OR dev$_m_net OR dev$_m_gen OR dev$_m_mbx OR dev$_m_dmt OR dev$_m_rtm);
488 0579 END
489 0580 ELSE
490 0581 BEGIN ! bits for "tapes"
491 0582 must_have = (dev$_m_sgd OR dev$_m_fod OR dev$_m_avl OR dev$_m_idv OR dev$_m_odv);
492 0583 cannot_have = (dev$_m_ccl OR dev$_m_trm OR dev$_m_spl OR dev$_m_opr
493 0584 OR dev$_m_net OR dev$_m_gen OR dev$_m_mbx OR dev$_m_dmt OR dev$_m_rtm);
494 0585 END;
495 0586
496 0587 ! If we are missing any 'must_have' items or if we have any 'cannot_have' items, scream and shout
497 0588
498 0589 IF (((.volb [volb$_devchar] XOR .must_have) AND .must_have) NEQ 0)
499 0590 OR
500 0591 ((.volb [volb$_devchar] AND .cannot_have) NEQ 0)
501 0592 THEN
502 0593 $exch_signal_return (exch$_devnotsuit, 1, dev_desc);
503 0594
504 0595 ! If the device is not mounted in the VMS sense, then we must do that
505 0596 ! and recursively call ourself
506 0597
507 0598 IF NOT .devbits [dev$_v_mnt]
508 0599 THEN
509 0600 BEGIN
510 0601 IF NOT exch$_moun_vms_mount (volb, dev_desc)
511 0602 THEN
512 0603 RETURN false;
513 0604 RETURN init_foreign_open ();
514 0605 END;
515 0606
516 0607 ! The device must be mounted foreign
517 0608
518 0609 IF NOT .devbits [dev$_v_for] ! If the volume is write-locked
519 0610 THEN
520 0611 $exch_signal_return (exch$_opnotperf11, 1, namb [namb$_q_device]);
521 0612
522 0613 END; ! scope 'devbits'
```

```
0614 2 ! Now set the unique ident field of this volb
0615 2
0616 2
0617 2 $debug_print_fao ('volb devnam "'AF'", namb device "'AF'", namb volid "'AF'", concealed !UL',
0618 2 .volb [volb$l_devnamlen], volb [volb$t_devnam],
0619 2 (BIND ndev = namb [namb$q_device] : $desc_block; .ndev [dsc$w_length]),
0620 2 (BIND ndev = namb [namb$q_device] : $desc_block; .ndev [dsc$a_pointer]),
0621 2 .namb [namb$l_vol_ident_len], namb [namb$t_vol_ident],
0622 2 .namb [namb$v_concealed_device]);
0623 2 CH$MOVE (volb$s_vol_ident, namb [namb$t_vol_ident], volb [volb$t_vol_ident]);
0624 2 volb [volb$l_vol_ident_len] = .namb [namb$l_vol_ident_len];
0625 2
0626 2 ! IF switch_debug ! Debugging trace code
0627 2 ! THEN
0628 2 BEGIN
0629 2 LOCAL
0630 2 tmp_desc : $desc_block;
0631 2 $stat_str_desc_init tmp_desc, .volb [volb$l_devnamlen], volb [volb$t_devnam];
0632 2 $debug_print_fao ('Getdvt for name "'AS'" resolved to device "'AS'", dev_desc, tmp_desc);
0633 2 END;
0634 2 ! FI
0635 2
0636 2 ! Init the RMS blocks for the volume
0637 2
0638 2 $fab_init (
0639 2 FAB = fab, ! Volume FAB
0640 2 FAC = (BIO,GET,PUT), ! Block I/O, read and write
0641 2 FNA = volb [volb$t_vol_ident], ! Set name addr
0642 2 FNS = .volb [volb$t_vol_ident_len], ! Set name size
0643 2 FOP = NFS, ! Non-File Structured
0644 2 NAM = nam); ! Name block
0645 2
0646 2 $rab_init (
0647 2 RAB = rab, ! Volume RAB
0648 2 ROP = BIO, ! Block I/O
0649 2 FAB = fab); ! FAB addr
0650 2
0651 2 $nam_init (
0652 2 NAM = nam, ! File name block
0653 2 RSA = .volb [volb$a_rsbuf], ! Result name addr
0654 2 RSS = nam$c_maxrss, ! Result name size
0655 2 ESA = .volb [volb$a_esbuf], ! Expanded name addr
0656 2 ESS = nam$c_maxrss); ! Expanded name size
0657 2
0658 2 ! Open and connect to the volume
0659 2
0660 2 IF NOT (status = $open (fab = fab))
0661 2 THEN
0662 2 RETURN exch$util_file_error (exch$_openforeign, .status, fab, .fab [fab$l_stv]);
0663 2
0664 2 volb [volb$w_channel] = .fab [fab$l_stv]; ! Save the channel number (NFS ==> user mode channel)
0665 2
0666 2 IF NOT (status = $connect (rab = rab))
0667 2 THEN
0668 2 RETURN exch$util_file_error (exch$_openforeign, .status, fab, .rab [rab$l_stv]);
0669 2
0670 2 ! Set the volume format
0671 2
0672 2 volb [volb$b_vol_format] = .namb [namb$b_vol_format];
```

```
.. 580      0671 2 volb [volb$y_vfmt_explicit] = .namb [namb$y_vfmt_explicit];
.. 581      0672 2 volb [volb$y_write] = (BIND devbits = fab [Tab$1_dev] : $bblock; (NOT .devbits [dev$y_sw]))); ! Device can
.. 582      0673 2
.. 583      0674 2 RETURN true;
.. 584      0675 1 END;
```

				OFFC 00000			
50	00000000G	EF	10	C1	00002		
		50	60	D0	0000A		
		59	60	D0	0000D		
		56	04	A0	D0	00010	
		57	10	A6	D0	00014	
		5A	14	A6	D0	00018	
		58	18	A6	D0	0001C	
		53	40	A9	9E	00020	
		52	002C00F9	8F	D0	00024	
		51	023B	8F	3C	0002B	
			00000000G	EF	16	00030	
		52	010A00F7	8F	D0	00036	
		51	023C	8F	3C	0003D	
		50		59	D0	00042	
			00000000G	EF	16	00045	
		52	041B00F3	8F	D0	0004B	
		51	023D	8F	3C	00052	
		50		56	D0	00057	
			00000000G	EF	16	0005A	
			0048	8F	BB	00060	
	00000000G	EF	02	FB	00064		
		5B	50	D0	0006B		
		17	5B	E8	0006E		
			0808	8F	BB	00071	
				01	DD	00075	
			00000000G	8F	DD	00077	
				04	FB	0007D	
	00000000G	00	5B	D0	00084		
		50		04	00087		
10	2F	A6	04	E1	00088	1\$:	
		50	1C054008	8F	D0	0008D	
		51	203220F7	8F	D0	00094	
				0E	11	0009B	
		50	0C044020	8F	D0	0009D	2\$:
		51	203220C6	8F	D0	000A4	
52	2C	A6	50	CD	000AB	3\$:	
		50		52	D3	000B0	
				06	12	000B3	
		51	2C	A6	D3	000B5	
				09	13	000B9	
		52	00000000G	8F	D0	000BB	4\$:
				27	11	000C2	
				.EXTRN	EXCH\$ ACCESSFAIL		
				.EXTRN	EXCH\$ DEVNOTSUIT		
				.EXTRN	EXCH\$ OPNOTPERF11		
				.EXTRN	SY\$OPEN, EXCH\$ OPENFOREIGN		
				.ENTRY	INIT FOREIGN OPEN, Save R2,R3,R4,R5,R6,R7,-	0500	
					R8,R9,R10,R11		
				ADDL3	#16, EXCH\$A_GBL, R0	0540	
				MOVL	(R0), R0	0541	
				MOVL	(R0), R9		
				MOVL	4(R0), R6	0542	
				MOVL	16(R6), R7	0543	
				MOVL	20(R6), R10	0544	
				MOVL	24(R6), R8	0545	
				MOVAB	64(R9), R3	0546	
				MOVL	#2883833, R2	0549	
				MOVZWL	#571, R1		
				JSB	EXCH\$UTIL_BLOCK_CHECK		
				MOVL	#17432823, R2	0550	
				MOVZWL	#572, R1		
				MOVL	R9, R0		
				JSB	EXCH\$UTIL_BLOCK_CHECK		
				MOVL	#68878579, R2	0551	
				MOVZWL	#573, R1		
				MOVL	R6, R0		
				JSB	EXCH\$UTIL_BLOCK_CHECK		
				PUSHR	#*M<R3,R6>	0555	
				CALLS	#2, EXCH\$UTIL_VOL_GETDVI		
				MOVL	R0, STATUS		
				BLBS	STATUS, 1\$		
				PUSHR	#*M<R3,R11>	0558	
				PUSHL	#1		
				PUSHL	#EXCH\$ ACCESSFAIL		
				CALLS	#4, LIB\$SIGNAL		
				MOVL	STATUS, R0	0559	
				RET			
				BBC	#4, 47(R6), 2\$	0573	
				MOVL	#470106120, MUST_HAVE	0576	
				MOVL	#540156151, CANNOT_HAVE	0577	
				BRB	3\$	0573	
				MOVL	#201605152, MUST_HAVE	0582	
				MOVL	#540156102, CANNOT_HAVE	0583	
				XORL3	MUST_HAVE, 44(R6), R2	0589	
				BITL	R2, MUST_HAVE		
				BNEQ	4\$		
				BITL	44(R6), CANNOT_HAVE	0591	
				BEQL	5\$		
				MOVL	#EXCH\$ DEVNOTSUIT, TEMP	0593	
				BRB	8\$		

17	2E	A6	03	E0	000C4	58:	BBS	#3, 46(R6), 78	0598		
			53	DD	000C9		PUSHL	R3	0601		
			56	DD	000CB		PUSHL	R6			
	00000000G	EF	02	FB	000CD		CALLS	#2, EXCH\$MOUN_VMS_MOUNT			
		03	50	EB	000D4		BLBS	R0, 68			
			00F4	31	000D7		BRW	138			
	FF21	CF	00	FB	000DA	68:	CALLS	#0, INIT_FOREIGN_OPEN	0604		
				04	000DF		RET				
		18	2F	A6	EB	000E0	78:	BLBS	47(R6), 98	0609	
		52	00000000G	8F	D0	000E4		MOVL	#EXCH\$_OPNOTPERF11, TEMP	0611	
				53	DD	000EB	88:	PUSHL	R3		
				01	DD	000ED		PUSHL	#1		
				52	DD	000EF		PUSHL	TEMP		
	00000000G	00		03	FB	000F1		CALLS	#3, LIB\$SIGNAL		
		50		52	D0	000F8		MOVL	TEMP, R0		
					04	000FB		RET			
69	A6	008A	C9	0080	8F	28	000FC	98:	MOV C3	#128, 138(R9), 105(R6)	0623
		65	A6	0086	C9	D0	00105		MOVL	134(R9), 101(R6)	0624
0050	8F	00	6E		00	2C	0010B		MOV C5	#0, (SP), #0, #80, (R7)	0644
					67		00112				
			67	5003	8F	B0	00113		MOVW	#20483, (R7)	
		04	A7	00010000	8F	D0	00118		MOVL	#65536, 4(R7)	
		16	A7		23	90	00120		MOVB	#35, 22(R7)	
		1F	A7		02	90	00124		MOVB	#2, 31(R7)	
		28	A7		58	D0	00128		MOVL	R8, 40(R7)	
		2C	A7	69	A6	9E	0012C		MOVAB	105(R6), 44(R7)	
0044	8F	00	34	65	A6	90	00131		MOVB	101(R6), 52(R7)	
			6E		00	2C	00136		MOV C5	#0, (SP), #0, #68, (R10)	0648
					6A		0013D				
			6A	4401	8F	B0	0013E		MOVW	#17409, (R10)	
		04	AA	0800	8F	3C	00143		MOVZWL	#2048, 4(R10)	
		3C	AA		57	D0	00149		MOVL	R7, 60(R10)	
0060	8F	00	6E		00	2C	0014D		MOV C5	#0, (SP), #0, #96, (R8)	0654
					68		00154				
			68	6002	8F	B0	00155		MOVW	#24578, (R8)	
		02	A8		01	8E	0015A		MNEGB	#1, 2(R8)	
		04	A8	20	A6	D0	0015E		MOVL	32(R6), 4(R8)	
		0A	A8		01	8E	00163		MNEGB	#1, 10(R8)	
		0C	A8	1C	A6	D0	00167		MOVL	28(R6), 12(R8)	
					57	DD	0016C		PUSHL	R7	0658
	00000000G	00			01	FB	0016E		CALLS	#1, SYSS\$OPEN	
		5B			50	D0	00175		MOVL	R0, STATUS	
		05			5B	EB	00178		BLBS	STATUS, 108	
			0C		A7	DD	0017B		PUSHL	12(R7)	0660
					17	11	0017E		BRB	118	
	4A	A6	0C		A7	B0	00180	108:	MOVW	12(R7), 74(R6)	0662
					5A	DD	00185		PUSHL	R10	0664
	00000000G	00			01	FB	00187		CALLS	#1, SYSS\$CONNECT	
		5B			50	D0	0018E		MOVL	R0, STATUS	
		15			5B	EB	00191		BLBS	STATUS, 128	
			0C		AA	DD	00194		PUSHL	12(R10)	0666
					57	DD	00197	118:	PUSHL	R7	
					5B	DD	00199		PUSHL	STATUS	
	00000000G	EF	00000000G		8F	DD	0019B		PUSHL	#EXCH\$ OPENFOREIGN	
					04	FB	001A1		CALLS	#4, EXCH\$UTIL_FILE_ERROR	
					04		001A8		RET		
	58	A6	7A	A9	90	001A9	128:	MOVB	122(R9), 88(R6)	0670	

EXCH\$INIT
V04-000

INIT verb dispatch and misc routines
init_foreign_open

L 7
16-Sep-1984 00:59:01
14-Sep-1984 12:29:05

VAX-11 Bliss-32 V4.0-742
[EXCHNG.SRC]EXC\$INIT.B32;1

Page 20
(6)

48	50	0085	C9	01	02	EF	001AE	EXTZV	#2, #1, 133(R9), R0	
	A6		01	06	50	F0	001B5	INSV	R0, #6, #1, 72(R6)	
	50	43	A7	01	01	EF	001BB	EXTZV	#1, #1, 67(R7), R0	
				50	50	D2	001C1	MCOML	R0, R0	
48	A6		01	05	50	F0	001C4	INSV	R0, #5, #1, 72(R6)	
				50	01	D0	001CA	MOVL	#1, R0	
						04	001CD	RET		
					50	D4	001CE	CLRL	R0	
						04	001D0	RET		

: 0671
: 0672
: 0674
: 0675

; Routine Size: 465 bytes, Routine Base: EXCH\$INIT_CODE + 0350

```
586 0676 1 GLOBAL ROUTINE init_init : NOVALUE = XSBTTL 'init_init'
587 0677 2 BEGIN
588 0678 3 ++
589 0679 4
590 0680 5 FUNCTIONAL DESCRIPTION:
591 0681 6
592 0682 7     Perform setups for EXCH$init_initialize
593 0683 8
594 0684 9 INPUTS:
595 0685 10
596 0686 11     none
597 0687 12
598 0688 13 IMPLICIT INPUTS:
599 0689 14
600 0690 15     global environment
601 0691 16
602 0692 17 OUTPUTS:
603 0693 18
604 0694 19     none
605 0695 20
606 0696 21 IMPLICIT OUTPUTS:
607 0697 22
608 0698 23     none
609 0699 24
610 0700 25 ROUTINE VALUE:
611 0701 26
612 0702 27     none
613 0703 28
614 0704 29 SIDE EFFECTS:
615 0705 30
616 0706 31     memory might be allocated for the init control block
617 0707 32
618 0708 33 --
619 0709 34 $dbgtrc_prefix ('init_init> ');
620 0710 35
621 0711 36 BIND
622 0712 37     init = exch$a_gbl [excg$a_init_work] : $ref_bblock ! pointer to our work area
623 0713 38     ;
624 0714 39
625 0715 40
626 0716 41 ! If our pointer is null, we need to allocate and initialize the work area
627 0717 42
628 0718 43 IF .init EQL 0
629 0719 44 THEN
630 0720 45     BEGIN
631 0721 46
632 0722 47         ! Get the right sized chunk of memory, conveniently set to nulls
633 0723 48
634 0724 49         init = exch$util_vm_allocate_zeroed (exchblk$s_init);
635 0725 50
636 0726 51         ! Set the ident fields
637 0727 52
638 0728 53         $block_init (.init, init);
639 0729 54
640 0730 55         ! Set the descriptors up
641 0731 56
642 0732 57         $dyn_str_desc_init (init [init$q_device]);
```



```
643 0733      $dyn_str_desc_init (init [init$q_volumeid]);
644 0734
645 0735      END;
646 0736
647 0737      ! Make sure that our work area is valid
648 0738
649 0739      $block_check (2, .init, init, 570);
650 0740
651 0741      RETURN;
652 0742      END;
```

```
53 00000000G EF 00000000G EF 001C 00000
63 10 C1 00009
63 D5 00011
22 12 00013
2C DD 00015
01 FB 00017
63 50 D0 0001E
OB A0 2C B0 00021
OA A0 07 8E 00025
50 63 0C C1 00029
60 64 7D 0002D
50 63 14 C1 00030
60 64 7D 00034
52 002C00F9 8F D0 00037 1$:
51 023A 8F 3C 0003E
50 00000000G EF 16 00046
04 0004C
```

.EXTRN EXCH\$GQ_DYN_STR_TEMPLATE

```
.ENTRY INIT_INIT, Save R2,R3,R4
MOVAB TMPL, R4
ADDL3 #16, EXCH$A_GBL, R3
TSTL (R3)
BNEQ 1$
PUSHL #44
CALLS #1, EXCH$UTIL_VM_ALLOCATE_ZEROED
MOVL R0, (R3)
MOVW #44, 8(R0)
MNEGB #7, 10(R0)
ADDL3 #12, (R3), R0
MOVQ TMPL, (R0)
ADDL3 #20, (R3), R0
MOVQ TMPL, (R0)
MOVL #2883833, R2
MOVZWL #570, R1
MOVL (R3), R0
JSB EXCH$UTIL_BLOCK_CHECK
RET
```

```
0676
0712
0718
0724
0728
0732
0733
0739
0742
```

; Routine Size: 77 bytes, Routine Base: EXCH\$INIT_CODE + 0521

```
654 0743 1 GLOBAL ROUTINE exch$init_initialize = %SBTTL 'exch$init_initialize'
655 0744 2 BEGIN
656 0745 3 ++
657 0746 4
658 0747 5 FUNCTIONAL DESCRIPTION:
659 0748 6
660 0749 7     Action routine for the INIT verb, parses and performs main control functions for INIT
661 0750 8
662 0751 9 INPUTS:
663 0752 10
664 0753 11     none
665 0754 12
666 0755 13 IMPLICIT INPUTS:
667 0756 14
668 0757 15     Command parameters and qualifiers as returned from CLISxxx routines.
669 0758 16
670 0759 17 OUTPUTS:
671 0760 18
672 0761 19     none
673 0762 20
674 0763 21 IMPLICIT OUTPUTS:
675 0764 22
676 0765 23     none
677 0766 24
678 0767 25 ROUTINE VALUE:
679 0768 26
680 0769 27     Success or worst error encountered.
681 0770 28
682 0771 29 SIDE EFFECTS:
683 0772 30
684 0773 31     Data is
685 0774 32
686 0775 33
687 0776 34 $dbgtrc_prefix ('init_initialize> ');
688 0777 35
689 0778 36 LOCAL
690 0779 37     message,
691 0780 38     namb      : $ref_bblock,      ! Local pointer to a namb
692 0781 39     volb      : $ref_bblock,      ! Local pointer to a volb
693 0782 40     status
694 0783 41 ;
695 0784 42
696 0785 43 BIND
697 0786 44     init = exch$a_gbl [excg$a_init_work] : $ref_bblock ! pointer to our work area
698 0787 45 ;
699 0788 46
700 0789 47 ! Allocate and/or initialize the work area
701 0790 48
702 0791 49 init_init ();
703 0792 50
704 0793 51 ! Get the individual boolean qualifiers.
705 0794 52
706 0795 53 init [init$v_q_create] = cli$present (%ASCII 'CREATE');
707 0796 54
708 0797 55
709 0798 56 ! Set the flag for printing init messages.
710 0799 57
```

```
711 0800 2 init [init$g_message] = .exch$a_gbl [exch$g_v_q_message];
712 0801 message = cli$present (%ASCII 'MESSAGE');
713 0802 IF .message EQL cli$present
714 0803 OR
715 0804 .message EQL cli$negated
716 0805 THEN
717 0806 init [init$g_q_message] = .message;
718 0807
719 0808 \\ init [init$g_q_badblocks] = cli$present (%ASCII 'BADBLOCKS');
720 0809 \\ init [init$g_q_badblocks_retain] = cli$present (%ASCII 'BADBLOCKS.RETAIN');
721 0810 \\ init [init$g_q_replace] = cli$present (%ASCII 'REPLACE');
722 0811 \\ init [init$g_q_replace_retain] = cli$present (%ASCII 'REPLACE.RETAIN');
723 0812
724 0813 ! Get individual integer-valued qualifiers, routine signals on errors. If the qualifier is not present, 0 if
725 0814 in the second parameter and -1 (success) is returned as the routine value. Here we also treat positionals
726 0815 second parameter as globals.
727 0816
728 0817 IF NOT (status = exch$cmd_cli_get_integer (%ASCII 'ALLOCATION', init [init$l_q_allocation]))
729 0818 THEN
730 0819 RETURN .status;
731 0820
732 0821 IF NOT (status = exch$cmd_cli_get_integer (%ASCII 'EXTRA_WORDS', init [init$l_q_extra_words]))
733 0822 THEN
734 0823 RETURN .status;
735 0824 IF .init [init$l_q_extra_words] GTRU 119
736 0825 THEN
737 0826 BEGIN
738 0827 $exch_signal (exch$rt11_extra);
739 0828 init [init$l_q_extra_words] = 119;
740 0829 END;
741 0830
742 0831 IF NOT (status = exch$cmd_cli_get_integer (%ASCII 'SEGMENTS', init [init$l_q_segments]))
743 0832 THEN
744 0833 RETURN .status;
745 0834 IF .init [init$l_q_segments] GTRU 31
746 0835 THEN
747 0836 BEGIN
748 0837 $exch_signal (exch$rt11_toomanyseg, 1, 31);
749 0838 init [init$l_q_segments] = 31;
750 0839 END;
751 0840
752 0841 ! Get the volume label
753 0842
754 0843 IF NOT (status = cli$get_value (%ASCII 'VOLUME LABEL', init [init$g_volumeid]))
755 0844 THEN
756 0845 $exch_signal_return (.status);
757 0846
758 0847 ! Parse the device name parameter into a newly allocated $NAMB, there are no defaults
759 0848
760 0849 status = exch$cmd_parse_filespec (%ASCII 'DEVICENAME', 0, 0, init [init$g_device], namb);
761 0850 init [init$a_namb] = .namb; ! Save it in the work area too
762 0851 IF NOT .status
763 0852 THEN
764 0853 $exch_signal_return (exch$parseerr, 1, init [init$g_device], .status);
765 0854
766 0855 ! If a physical init, check the name
767 0856
```



```
768 0857 IF NOT (.init [init$v_q_create])
769 0858 THEN
770 0859 BEGIN
771 0860 IF NOT .namb [namb$v_explicit_device]
772 0861 THEN
773 0862   $exch_signal_return (exch$nodevice, 1, init [init$q_device]);
774 0863 IF .namb [namb$v_explicit_node]
775 0864 THEN
776 0865   $exch_signal_return (exch$nodevice, 1, init [init$q_device]);
777 0866 IF .namb [namb$v_explicit_directory] OR .namb [namb$v_explicit_name]
778 0867 OR .namb [namb$v_explicit_type] OR .namb [namb$v_explicit_version]
779 0868 THEN
780 0869   $exch_signal (exch$_devonly, 1, init [init$q_device]);
781 0870 END;
782 0871
783 0872 ! If the device is not mounted, attempt to temporarily open a file and perform the operation
784 0873
785 0874 volb = .namb [namb$a_assoc_volb]; ! If it is mounted, we will have a pointer to a volb
786 0875 IF (.volb EQL 0)
787 0876 THEN
788 0877 BEGIN
789 0878   ! Allocate a $VOLB to describe the volume
790 0879   !
791 0880   volb = exch$util_volb_allocate ();
792 0881   init [init$a_volb] = .volb;
793 0882   !
794 0883   ! Temporarily open a channel to the device
795 0884   !
796 0885   IF .init [init$v_q_create]
797 0886 THEN
798 0887     status = init_foreign_create ();
799 0888 ELSE
800 0889     status = init_foreign_open ();
801 0890   !
802 0891   ! Now do the actual initialize
803 0892   !
804 0893   IF .status
805 0894 THEN
806 0895     BEGIN
807 0896       ! The open worked, let's see if we can do the volume-specific part of it
808 0897       !
809 0898       !
810 0899       CASE .volb [volb$b_vol_format] FROM volb$k_vfmt_lobound TO volb$k_vfmt_hibound OF
811 0900       SET
812 0901         [volb$k_vfmt_dos11] : BEGIN
813 0902           status = init_dos11_init ();
814 0903           CHSMOVE (6, UPLIT BYTE ('DOS-11'), volb [volb$t_vol_type]);
815 0904           volb [volb$l_vol_type_len] = 6;
816 0905           END;
817 0906         [volb$k_vfmt_rt11] : BEGIN
818 0907           status = init_rt11_init ();
819 0908           CHSMOVE (5, UPLIT BYTE ('RT-11'), volb [volb$t_vol_type]);
820 0909           volb [volb$l_vol_type_len] = 5;
821 0910           END;
822 0911         [volb$k_vfmt_rmt] : $exch_signal_stop (exch$_notimplement);
823 0912         [OUTRANGE, INRANGE] : $logic_check (0, (false), 226);
824 0913
```

```
00000000 0914 4      TES;
00000001 0915 4
00000002 0916 4      ! Close the volb's file now
00000003 0917 4
00000004 0918 4      ! init_foreign_close ();
00000005 0919 4      END;
00000006 0920 4
00000007 0921 4      ! Release the volb, since we don't plan to mount it
00000008 0922 4
00000009 0923 4      exch$util_volb_release (.volb);
00000010 0924 4
00000011 0925 4      END
00000012 0926 4
00000013 0927 4      ! OK, the device has already been mounted
00000014 0928 4
00000015 0929 4      ELSE
00000016 0930 4      BEGIN
00000017 0931 4
00000018 0932 4      ! The open worked, let's see if we can do the volume-specific part of it
00000019 0933 4
00000020 0934 4      ! init [init$a_volb] = .volb;
00000021 0935 4      CASE .volb [volb$b_vol_format] FROM volb$k_vfmt_lobound TO volb$k_vfmt_hibound OF
00000022 0936 4      SET
00000023 0937 4          [volb$k_vfmt_dos11] : status = init_dos11_init ();
00000024 0938 4          [volb$k_vfmt_rt11] : status = init_rt11_init ();
00000025 0939 4      !\ [volb$k_vfmt_rtmt] : $exch_signal_stop ($exch$notimplement);
00000026 0940 4          [OUTRANGE, INRANGE] : $logic_check (0, (false), 307);
00000027 0941 4      TES;
00000028 0942 4
00000029 0943 4      END;
00000030 0944 4
00000031 0945 4      ! Tell them it has been done
00000032 0946 4
00000033 0947 4      IF .status
00000034 0948 4      AND
00000035 0949 4      ! init [init$v_q_message]
00000036 0950 4      THEN
00000037 0951 4      P $exch_signal (exch$initialized, 4, .volb [volb$l_vol_type_len], volb [volb$t_vol_type],
00000038 0952 4      ! .volb [volb$l_vol_ident_len], volb [volb$t_vol_ident]);
00000039 0953 4
00000040 0954 4      ! Release the namb we used for the input
00000041 0955 4
00000042 0956 4      exch$util_namb_release (.namb);
00000043 0957 4
00000044 0958 4      RETURN .status;
00000045 0959 4      END;
```

.PSECT EXCH\$INIT_PLIT, NOWRT, 2

```
00 00 45 54 41 45 52 43 0001B .BLKB 1
0001C P.AAE: .ASCII \CREATE\<0><0>
010E0006 00024 P.AAD: .LONG 17694726
00000000 00028 .ADDRESS P.AAE
00 45 47 41 53 53 45 4D 0002C P.AAG: .ASCII \MESSAGE\<0>
010E0007 00034 P.AAF: .LONG 17694727
```

```
00 00 4E 4F 49 54 41 43 4F 4C 00000000' 00038
                                4C 41 0003C P.AAI:
                                010E000A 00048 P.AAH:
00 53 44 52 4F 57 5F 41 52 54 00000000' 0004C
                                58 45 00050 P.AAK:
                                010E000B 0005C P.AAJ:
                                00000000' 00060
                                53 54 4E 45 4D 47 45 53 00064 P.AAM:
                                010E000B 0006C P.AAL:
00 4C 45 42 41 4C 45 4D 55 4C 00000000' 00070
                                4F 56 00074 P.AAO:
                                010E000B 00080 P.AAN:
00 00 45 4D 41 4E 45 43 49 56 00000000' 00084
                                45 44 00088 P.AAQ:
                                010E000A 00094 P.AAP:
                                00000000' 00098
                                31 31 2D 53 4F 44 0009C P.AAR:
                                31 31 2D 54 52 000A2 P.AAS:
```

```
.ADDRESS P.AAG
.ASCII \ALLOCATION\<0><0>
.LONG 17694730
.ADDRESS P.AAI
.ASCII \EXTRA WORDS\<0>
.LONG 17694731
.ADDRESS P.AAK
.ASCII \SEGMENTS\
.LONG 17694728
.ADDRESS P.AAM
.ASCII \VOLUME LABEL\<0>
.LONG 17694731
.ADDRESS P.AAO
.ASCII \DEVICENAME\<0><0>
.LONG 17694730
.ADDRESS P.AAQ
.ASCII \DOS-11\
.ASCII \RT-11\
```

```
.EXTRN CLIS_PRESENT, CLIS_NEGATED
.EXTRN EXCH$RT11_EXTRA
.EXTRN EXCH$RT11_TOOMANYSEG
.EXTRN CLISGET_VALUE, EXCH$PARSEERR
.EXTRN EXCH$NODEVICE, EXCH$NOREMOTE
.EXTRN EXCH$DEVONLY, EXCH$BADLOGIC
.EXTRN EXCH$INITIALIZED
```

```
.PSECT EXCH$INIT_CODE, NOWRT, 2
```

```
.ENTRY EXCH$INIT_INITIALIZE, Save R2,R3,R4,R5,R6,- 0743
R7,R8,R9,R10,R11
```

```
MOVAB P.AAD, R11
MOVAB LIB$SIGNAL, R10
SUBL2 #4, SP
ADDL3 #16, EXCH$A_GBL, R2
CALLS #0, INIT_INIT
MOVL (R2), R2
MOVAB 40(R2), R9
PUSHL R11
CALLS #1, CLISPRESENT
INSV R0, #0, #1, (R9)
EXTZV #2, #1, @EXCH$A_GBL, R0
INSV R0, #1, #1, (R9)
PUSHAB P.AAF
CALLS #1, CLISPRESENT
CMPL MESSAGE, #CLIS_PRESENT
BEQL 1$
CMPL MESSAGE, #CLIS_NEGATED
BNEQ 2$
INSV MESSAGE, #1, #1, (R9)
PUSHAB 28(R2)
PUSHAB P.AAH
CALLS #2, EXCH$CMD_CLI_GET_INTEGER
MOVL R0, STATUS
BLBC STATUS, 4$
PUSHAB 32(R2)
PUSHAB P.AAJ
```

0786
0792
0796

0800

0801

0802

0804

0806

0817

0821

```
OFFC 00000
5B 0000' CF 9E 00002
5A 00000000G 00 9E 00007
5E 04 C2 0000E
52 00000000G EF 10 C1 00011
96 AF 00 FB 00019
52 62 D0 0001D
59 28 A2 9E 00020
5B DD 00024
00 01 FB 00026
00 50 F0 0002D
50 00000000G FF 01 02 EF 00032
69 01 01 50 F0 0003B
01 10 AB 9F 00040
00000000G 00 01 FB 00043
00000000G 8F 50 D1 0004A
09 13 00051
0000G000G BF 50 D1 00053
05 12 0005A
69 01 01 50 F0 0005C 1$:
1C A2 9F 00061 2$:
24 AB 9F 00064
02 FB 00067
50 D0 0006E
58 E9 00071
20 A2 9F 00074
38 AB 9F 00077
```


00000000G	EF	02	FB	0007A	CALLS	#2, EXCH\$CMD_CLI_GET_INTEGER	
	58	50	DO	00081	MOVL	R0, STATUS	
	28	58	E9	00084	BLBC	STATUS, 4\$	
00000077	8F	20	A2	D1	00087	CMPL	32(R2), #119
			0E	1B	0008F	BLEQU	3\$
		00000000G	8F	DD	00091	PUSHL	#EXCH\$ RT11 EXTRA
	6A		01	FB	00097	CALLS	#1, LIB\$SIGNAL
20	A2	77	8F	9A	0009A	MOVZBL	#119, 32(R2)
		24	A2	9F	0009F	PUSHAB	36(R2)
		48	AB	9F	000A2	PUSHAB	P.AAL
00000000G	EF		02	FB	000A5	CALLS	#2, EXCH\$CMD_CLI_GET_INTEGER
	58		50	DO	000AC	MOVL	R0, STATUS
	03		58	E8	000AF	BLBS	STATUS, 5\$
		0191	31	000B2	BRW	31\$	
	1F	24	A2	D1	000B5	CMPL	36(R2), #31
			11	1B	000B9	BLEQU	6\$
			1F	DD	000BB	PUSHL	#31
			01	DD	000BD	PUSHL	#1
		00000000G	8F	DD	000BF	PUSHL	#EXCH\$ RT11 TOOMANYSEG
	6A		03	FB	000C5	CALLS	#3, LIB\$SIGNAL
24	A2		1F	DO	000C8	MOVL	#31, 36(R2)
		14	A2	9F	000CC	PUSHAB	20(R2)
		5C	AB	9F	000CF	PUSHAB	P.AAN
00000000G	00		02	FB	000D2	CALLS	#2, CLISGET_VALUE
	58		50	DO	000D9	MOVL	R0, STATUS
	0A		58	E8	000DC	BLBS	STATUS, 7\$
	53		58	DO	000DF	MOVL	STATUS, TEMP
			53	DD	000E2	PUSHL	TEMP
	6A		01	FB	000E4	CALLS	#1, LIB\$SIGNAL
			32	11	000E7	BRB	8\$
			5E	DD	000E9	PUSHL	SP
	54	0C	A2	9E	000EB	MOVAB	12(R2), R4
			54	DD	000EF	PUSHL	R4
			7E	7C	000F1	CLRQ	-(SP)
		70	AB	9F	000F3	PUSHAB	P.AAP
00000000G	EF		05	FB	000F6	CALLS	#5, EXCH\$CMD_PARSE_FILESPEC
	58		50	DO	000FD	MOVL	R0, STATUS
	57		6E	DO	00100	MOVL	NAMB, R7
	62		57	DO	00103	MOVL	R7, (R2)
	16		58	E8	00106	BLBS	STATUS, 9\$
	53	00000000G	8F	DO	00109	MOVL	#EXCH\$_PARSEERR, TEMP
		0110	8F	BB	00110	PUSHR	#*M<R4,R8>
			01	DD	00114	PUSHL	#1
			53	DD	00116	PUSHL	TEMP
	6A		04	FB	00118	CALLS	#4, LIB\$SIGNAL
	50		53	DO	0011B	MOVL	TEMP, R0
			04	0011E	RET		
	46		69	E8	0011F	BLBS	(R9), 14\$
	53	6C	A7	9E	00122	MOVAB	108(R7), R3
			63	95	00126	TSTB	(R3)
			09	19	00128	BLSS	10\$
	55	00000000G	8F	DO	0012A	MOVL	#EXCH\$_NODEVICE, TEMP
			0B	11	00131	BRB	11\$
14	63		06	E1	00133	BBC	#6, (R3), 12\$
	55	00000000G	8F	DO	00137	MOVL	#EXCH\$_NOREMOTE, TEMP
			54	DD	0013E	PUSHL	R4
			01	DD	00140	PUSHL	#1

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

EXCH\$INIT
V04-000

INIT verb dispatch and misc routines
exch\$init_initialize

1 8
16-Sep-1984 00:59:01
14-Sep-1984 12:29:05

VAX-11 Bliss-32 V4.0-742
[EXCHNG.SRC]EXCINIT.B32;1

Page 30
(8)

F87D	CF	00	FB	00210	268:	CALLS	#0, INIT_DOS11_INIT	0937
		05	11	00215		BRB	288	
0000V	CF	00	FB	00217	278:	CALLS	#0, INIT_RT11_INIT	0938
	58	50	DD	0021C	288:	MOVL	R0, STATUS	
	1B	58	E9	0021F	298:	BLBC	STATUS, 308	0947
17	69	01	E1	00222		BBC	#1, (R9), 308	0949
		69	A6	9F	00226	PUSHAB	105(VOLB)	0952
		65	A6	DD	00229	PUSHL	101(VOLB)	
		5D	A6	9F	0022C	PUSHAB	93(VOLB)	
		59	A6	DD	0022F	PUSHL	89(VOLB)	
			04	DD	00232	PUSHL	#4	
	00000000G	8F	DD	00234		PUSHL	#EXCH\$ INITIALIZED	
	6A	06	FB	0023A		CALLS	#6, LIB\$SIGNAL	
		57	DD	0023D	308:	PUSHL	R7	0956
00000000G	EF	01	FB	0023F		CALLS	#1, EXCH\$UTIL_NAMB_RELEASE	
	50	58	DD	00246	318:	MOVL	STATUS, R0	0958
		04	00249			RET		0959

; Routine Size: 586 bytes. Routine Base: EXCH\$INIT_CODE + 056E


```
0960 1 GLOBAL ROUTINE init_rt11_init = %SBTTL 'init_rt11_init'
0961 BEGIN
0962 ++
0963
0964 FUNCTIONAL DESCRIPTION:
0965
0966     Perform RT11 volume specific init actions
0967
0968 INPUTS:
0969
0970     none
0971
0972 IMPLICIT INPUTS:
0973
0974     work area for INIT
0975
0976 OUTPUTS:
0977
0978     none
0979
0980 IMPLICIT OUTPUTS:
0981
0982     none
0983
0984 ROUTINE VALUE:
0985
0986     Success or worst error encountered.
0987
0988 SIDE EFFECTS:
0989
0990     RT11 directory will be initialized
0991
0992 --
0993 $dbgtrc_prefix ('init_rt11_init> ');
0994
0995 LOCAL
0996     ent : $ref_bblock,      | the first entry in the block
0997     hdr : $ref_bblock,      | pointer to the rt11 directory block
0998     hom : $ref_bblock,      | pointer to the rt11 home block
0999     rtv : $ref_bblock,      | rt11 volume extension
1000     bnum,                    | number of blocks on device
1001     snum,                    | number of segments in directory
1002     start,                   | start block for files
1003     hdrbuf : $bvector [rt11$k_dirseglen],
1004     status                    | actual buffer
1005     ;
1006
1007 BIND
1008     init = exch$a_gbl [excg$a_init_work] : $ref_bblock, | pointer to our work area
1009     volb = init [init$a_volb] : $ref_bblock | pointer to exchange VOLB structure
1010     ;
```

```
924 1011 2 ! Boot program. The following PDP-11 program will type out the attached message when the volume is booted on
925 1012 ! PDP-11, informing the user that this is not a system disk. (Thanks to <INIT.SRC>ININDX.B32)
926 1013
927 1014 BIND
928 1015 boot_program = UPLIT WORD (
929 1016
930 1017 %0'000240',
931 1018 %0'012706', %0'001000',
932 1019 %0'010700',
933 1020 %0'062700', %0'000036',
934 1021 %0'112001',
935 1022 %0'001403',
936 1023 %0'004767', %0'000006',
937 1024 %0'000773',
938 1025 %0'000005',
939 1026 %0'000000',
940 1027
941 1028
942 1029 %0'110137', %0'177566',
943 1030 %0'105737', %0'177564',
944 1031 %0'100375',
945 1032 %0'000207',
946 1033
947 1034
948 1035
949 1036 ),
950 1037
951 1038 ! Boot message, we will add the volume id a little later
952 1039
953 1040 boot_message = UPLIT BYTE (
954 1041 7, 13, 10, 10, 7,
955 1042 'The volume labeled "
956 1043 7, 13, 10, 10, 7, 0
957 1044 );
958 1045
959 1046 LITERAL
960 1047 boot_prog_len = 38, ! boot program is 38 bytes long
961 1048 boot_mesg_len = 68, ! message is 68 bytes long
962 1049 boot_volname = boot_prog_len+25; ! volume label offset in boot block message
```

```
BOOTBK: NOP
MOV #1000,SP
PC,R0
MOV #BOTMSG-.,R0
10$: MOVB (R0)+,R1
BEQ 20$
CALL TYPIT
BR 10$
20$: RESET
HALT

TYPIT: MOVB R1,@#TPB
10$: TSTB @#TPS
BPL 10$
RETURN

BOTMSG:
```

```
: NOP IDENTIFIES BOO
: SET TEMP STACK
: SET ADDRESS
: OF MESSAGE
: GET NEXT CHARACTER
: END
: NO, PRINT IT
: LOOP FOR NEXT CHAR
:
:
: PRINT CHARACTER
: DONE?
: NO, WAIT
:
```

```

964 1050 2 $block_check (2, .init, init, 574);
965 1051 2 $block_check (2, .volb, volb, 576);
966 1052 2
967 1053 2 ! Make sure that we can do it
968 1054 2
969 1055 2 IF NOT .volb [volb$v_write]
970 1056 2 THEN
971 P 1057 2   $exch_signal_return ($warning_stat copy (exch$ writelock), 2,
972 1058 2   .volb [volb$l_vol_tident_len], volb [volb$t_vol_ident]);
973 1059 2
974 1060 2 ! Get a zeroed buffer for the block and a pointer to the first entry
975 1061 2
976 1062 2   hdr = hdrbuf;
977 1063 2   hom = hdrbuf + 512;
978 1064 2   CH$FILL (0, rt11$k_dirseglen, hdrbuf);
979 1065 2   ent = .hdr + rt11hdr$k_length;
980 1066 2
981 1067 2 ! Determine the number of device blocks
982 1068 2
983 1069 2   bnum = (BEGIN
984 1070 2     LOCAL
985 1071 2     bmax;
986 1072 2     bmax = MINU (65535, .volb [volb$l_devmaxblock]);
987 1073 2     IF .volb [volb$v_virtual]
988 1074 2     THEN
989 1075 2       BEGIN
990 1076 2         IF .init [init$l_q_allocation] NEQ 0
991 1077 2         AND
992 1078 2         NOT .init [init$v_q_create]
993 1079 2         THEN
994 1080 2           $exch_signal (exch$_virtnochange);
995 1081 2           .bmax
996 1082 2         END
997 1083 2     ELSE IF .init [init$l_q_allocation] NEQ 0
998 1084 2     THEN
999 1085 2       BEGIN
1000 1086 2         IF .init [init$l_q_allocation] GTRU .bmax
1001 1087 2         THEN
1002 1088 2           BEGIN
1003 1089 2             $exch_signal (exch$_rt11_toomanyblk, 1, .bmax);
1004 1090 2             .bmax
1005 1091 2           END
1006 1092 2         ELSE
1007 1093 2           .init [init$l_q_allocation]
1008 1094 2         END
1009 1095 2     ELSE
1010 1096 2       .bmax
1011 1097 2     END);
1012 1098 2   bnum = MAXU (40, .bnum);
```

```
1014 1099 2 ! Determine the number of directory segments
1015 1100
1016 1101 snum = (SELECTONE true OF
1017 1102 SET
1018 1103
1019 1104 ! If a /SEGMENTS was given, use that value
1020 1105 [.init [init$L_q_segments] NEQ 0] : .init [init$L_q_segments];
1021 1106
1022 1107 ! If no /SEGMENTS, use a default based on device size (ala RT-11 DUP)
1023 1108
1024 1109
1025 1110 [.bnum LEQU 512] : 1;
1026 1111 [.bnum LEQU 2048] : 4;
1027 1112 [.bnum LEQU 12288] : 16;
1028 1113 [OTHERWISE] : 31;
1029 1114
1030 1115 TES);
1031 1116
1032 1117 ! Determine the start block for files
1033 1118
1034 1119 start = rt11$k_root_block + (2 * .snum);
1035 1120 IF .start+32 GTRU .bnum ! If room for fewer than 32 blocks for files
1036 1121 THEN
1037 1122 BEGIN
1038 1123 snum = 1; ! Reduce to one segment
1039 1124 start = rt11$k_root_block + 2; ! Start at a given block
1040 1125 $exch_signal (@exch$rt11_toomanyseg, 1, 1); ! And tell the world
1041 1126 END;
```



```
1043 1127 2 ! Set up the boot and home blocks
1044 1128 2
1045 L 1129 2 $logic_check (0, (rt11hom$owner_name EQL excg$username), 310);
XPRINT: assumption 310-verified during compilation
1046 1130 2 CH$MOVE (rt11hom$owner_name, excg$gbl [excg$username], hom [rt11hom$owner_name]);
1047 1131 2 CH$MOVE (rt11hom$system_id, UPLIT BYTE ('DECVMSXCHNG'), hom [rt11hom$system_id]);
1048 1132 2 CH$MOVE (boot_prog_len + boot_mesg_len, boot_program, hdrbuf [0]);
1049 1133 4 (BEGIN
1050 1134 4 BIND
1051 1135 4 desc = init [init$q_volumeid] : $desc block;
1052 1136 4 CH$COPY (.desc [dsc$w_length], .desc [dsc$a_pointer], %C ' ', rt11hom$volume_id, hom [rt11hom$volume_id
1053 1137 4 CH$COPY (.desc [dsc$w_length], .desc [dsc$a_pointer], %C ' ', rt11hom$volume_id, hdrbuf [boot_volname]);
1054 1138 2 END);
1055 1139 2 hom [rt11hom$w_system_vers] = %RAD50_11 'V40';
1056 1140 2 hom [rt11hom$w_cluster] = 1;
1057 1141 2 hom [rt11hom$w_first_seg] = rt11$k_root_block;
1058 1142 2
1059 1143 2 ! Write the boot and home blocks.
1060 1144 2
1061 1145 2 IF NOT (status = excg$io_rt11_write (.volb, 0, 2, .hdr))
1062 1146 2 THEN
1063 1147 2 RETURN .status;
1064 1148 2
1065 1149 2 ! If the volume format extension exists, overwrite the cached home block
1066 1150 2
1067 1151 2 rtv = .volb [volb$a_vfmt_specific];
1068 1152 2 IF .rtv NEQ 0
1069 1153 2 THEN
1070 1154 2 BEGIN
1071 1155 2 $block check (2, .rtv, rt11, 629);
1072 1156 2 CH$MOVE (512, .hom, rtv [rt11$block_1]);
1073 1157 2 END;
1074 1158 2
1075 1159 2 ! We will zero the disk to the end of the directory area.
1076 1160 2
1077 1161 2 CH$FILL (0, rt11$k_dirseglen, hdrbuf);
1078 1162 2 INCR p FROM 2 TO .start-1 BY 2
1079 1163 2 DO
1080 1164 2 IF NOT (status = excg$io_rt11_write (.volb, .p, 2, .hdr))
1081 1165 2 THEN
1082 1166 2 RETURN .status;
1083 1167 2
1084 1168 2 ! Since Files-11 writes a large number of home blocks on a device, make sure that we zero most of them so th
1085 1169 2 don't see strange things happening during a foreign mount.
1086 1170 2
1087 1171 2 IF NOT (status = init_zero_home_blocks (.start, .hdr))
1088 1172 2 THEN
1089 1173 2 RETURN .status;
1090 1174 2
1091 1175 2 ! Now set up the header of the first segment
1092 1176 2
1093 1177 2 hdr [rt11hdr$w_num_segs] = .snum;
1094 1178 2 hdr [rt11hdr$w_next_seg] = 0;
1095 1179 2 hdr [rt11hdr$w_high_seg] = 1;
1096 1180 2 hdr [rt11hdr$w_extra_bytes] = 2 * .init [init$l_q_extra_words];
1097 1181 2 hdr [rt11hdr$w_start_block] = .start;
1098 1182 2
```

! If not an rtv we are hopelessly co
! Copy the home block to cache

! Pass # of first unzeroed block and zeroed

! Only one segment in the directory

```
1099 1183 2 ! Make the empty entry followed by end of segment marker
1100 1184 2 !
1101 1185 2 ent [rt11ent$b_type_byte] = rt11ent$m_typ_empty;
1102 1186 2 ent [rt11ent$l_filename] = r50_empty; ! Name is simple 'EMPTY.FIL'
1103 1187 2 ent [rt11ent$w_filetype] = r50_fil;
1104 1188 2 exch$rt11_format_current_date (.ent);
1105 1189 2 ent [rt11ent$w_blocks] = .bnum - .hdr [rt11hdr$w_start_block];
1106 1190 2 ent = .ent + rt11ent$k_length + .hdr [rt11hdr$w_extra_bytes];
1107 1191 2 $logic_check (2, (.ent-LSSU .hdr + 510), 247);
1108 1192 2 ent [rt11ent$b_type_byte] = rt11ent$m_typ_end_segment;
1109 1193 2
1110 1194 2 ! If the volume format extension exists, overwrite the cached directory
1111 1195 2
1112 1196 2 IF .rtv NEQ 0
1113 1197 2 THEN
1114 1198 2 BEGIN
1115 1199 2 CH$MOVE (512, .hdr, rtv [rt11st_dire_segments]); ! Copy the new directory to cache
1116 1200 2 $logic_check (2, (exch$rtacp_verify_directory (.volb)), 249); ! Make sure the directory is still o
1117 1201 2 END;
1118 1202 2
1119 1203 2 ! Write out the new root directory, only the first block necessary
1120 1204 2
1121 1205 2 status = exch$io_rt11_write (.volb, rt11$k_root_block, 1, .hdr);
1122 1206 2
1123 1207 2 RETURN .status;
1124 1208 2 1 END;
```

```
0006 09F7 0303 9401 001E 65C0 11C0 0200 15C6 00A0 000A7 P.AAT: .BLKB 1
0087 80FD FF74 8BDF FF76 905F 0000 0005 01FB 000AB .WORD 160, 5574, 512, 4544, 26048, 30, -27647, -
000BC 000BC 771, 2551, 6, 507, 5, 0, -28577, -138, -
000CE P.AAU: .BYTE 7, 13, 10, 10, 7
000D3 .ASCII \the volume labeled " " is not\
000E2
000F1
000FB .ASCII \ a system volume.\
0010A
0010C P.AAV: .BYTE 7, 13, 10, 10, 7, 0
00112 .ASCII \DECVMSXCHNG\
00112
BOOT_PROGRAM= P.AAT
BOOT_MESSAGE= P.AAU
.EXTRN EXCH$_VIRTNOCHANGE
.PSECT EXCH$INIT_CODE,NOWRT,2
.OFFC 00000 .ENTRY INIT_RT11_INIT, Save R2,R3,R4,R5,R6,R7,R8,- : 0960
R9,R10,R11
MOVAB -1044(SP), SP
ADDL3 #16, EXCH$_GBL, R0
MOVL (R0), R10
MOVL #2883833, R2
MOVZWL #574, R1 : 1008
: 1009
: 1050
```

EXC
V04

		04	AE	24	AA	DO	00108	MOVL	36(R10), SNUM		
					31	11	0010D	BRB	11\$		
		00000200	8F		56	D1	0010F	7\$:	CMPL	BNUM, #512	1110
					06	1A	00116	BGTRU	8\$		
		04	AE		01	DO	00118	MOVL	#1, SNUM		
					22	11	0011C	BRB	11\$		
		00000800	8F		56	D1	0011E	8\$:	CMPL	BNUM, #2048	1111
					06	1A	00125	BGTRU	9\$		
		04	AE		04	DO	00127	MOVL	#4, SNUM		
					13	11	00128	BRB	11\$		
		00003000	8F		56	D1	0012D	9\$:	CMPL	BNUM, #12288	1112
					06	1A	00134	BGTRU	10\$		
		04	AE		10	DO	00136	MOVL	#16, SNUM		
					04	11	0013A	BRB	11\$		
		04	AE		1F	DO	0013C	10\$:	MOVL	#31, SNUM	1113
OC	AE	04	AE		01	78	00140	11\$:	ASHL	#1, SNUM, START	1119
		OC	AE		06	C0	00146	ADDL2	#6, START		
	50	OC	AE		20	C1	0014A	ADDL3	#32, START, R0		1120
			56		50	D1	0014F	CMPL	R0, BNUM		
					19	1B	00152	BLEQU	12\$		
		04	AE		01	DO	00154	MOVL	#1, SNUM		1123
		OC	AE		08	DO	00158	MOVL	#8, START		1124
					01	DD	0015C	PUSHL	#1		1125
					01	DD	0015E	PUSHL	#1		
				00000000G	8F	DD	00160	PUSHL	#EXCH\$ RT11, TOOMANYSEG		
		00000000G	00		03	FB	00166	CALLS	#3, LIB\$SIGNAL		
			50	00000000G	EF	DO	0016D	12\$:	MOVL	EXCH\$A GBL, R0	1130
01E4	CB	20	AO		OC	28	00174	MOV C3	#12, 32(R0), 484(HOM)		
01F0	CB	0000'	CF		OC	28	00178	MOV C3	#12, P.AAV, 496(HOM)		1131
14	AE	0000'	CF	006A	8F	28	00183	MOV C3	#106, BOOT PROGRAM, HDRBUF		1132
			57	14	AA	9E	0018C	MOVAB	20(R10), R7		1135
OC	20	04	B7		67	2C	00190	MOV C5	(R7), #4(R7), #32, #12, 472(HOM)		1136
				01D8	CB		00196				
OC	20	04	B7		67	2C	00199	MOV C5	(R7), #4(R7), #32, #12, HDRBUF+63		1137
				53	AE		0019F				
		01D2	CB		01	B0	001A1	MOVW	#1, 466(HOM)		1140
		01D4	CB	8EEEE0006	8F	DO	001A6	MOVL	#-1897005050, 468(HOM)		1141
					58	DD	001AF	PUSHL	HDR		1145
					02	DD	001B1	PUSHL	#2		
				OC	7E	D4	001B3	CLRL	-(SP)		
					AE	DD	001B5	PUSHL	12(SP)		
		00000000G	EF		04	FB	001B8	CALLS	#4, EXCH\$IO_RT11_WRITE		
			AE		50	DO	001BF	MOVL	R0, STATUS		
		08	70		AE	E9	001C3	BLBC	STATUS, 16\$		
			6E	00000054	8F	C1	001C7	ADDL3	#84, (SP), R0		1151
			57		60	DO	001CF	MOVL	(R0), RTV		
				10	AE	D4	001D2	CLRL	16(SP)		1152
					57	D5	001D5	TSTL	RTV		
					20	13	001D7	BEQL	13\$		
				10	AE	D6	001D9	INCL	16(SP)		
			52	880E00F5	8F	DO	001DC	MOVL	#-2012348171, R2		1155
			51	0275	8F	3C	001E3	MOVZWL	#629, R1		
			50		57	DO	001E8	MOVL	RTV, R0		
				00000000G	EF	16	001EB	JSB	EXCH\$UTIL BLOCK CHECK		
				0200	8F	28	001F1	MOV C3	#512, (HOM), 528(RTV)		1156
0400	8F	020E	C7		00	2C	001F9	13\$:	MOV C5	#0, (SP), #0, #1024, HDRBUF	1161
			00		AE		00200				
				14							

	53	OC	AE	01	C3	00202	SUBL3	#1, START, R3	1162
				52	D4	00207	CLRL	P	
				18	11	00209	BRB	15\$	
				58	DD	0020B	PUSHL	HDR	1164
				02	DD	0020D	PUSHL	#2	
				52	DD	0020F	PUSHL	P	
			OC	AE	DD	00211	PUSHL	12(SP)	
	00000000G	EF		04	FB	00214	CALLS	#4, EXCH\$IO_RT11_WRITE	
	08	AE		50	DD	0021B	MOVL	R0, STATUS	
FFE2		14	08	AE	E9	0021F	BLBC	STATUS, 16\$	
	52	02		53	F1	00223	ACBL	R3, #2, P, 14\$	
				58	DD	00229	PUSHL	HDR	1171
			10	AE	DD	0022B	PUSHL	START	
	0000V	CF		02	FB	0022E	CALLS	#2, INIT_ZERO_HOME_BLOCKS	
	08	AE		50	DD	00233	MOVL	R0, STATUS	
		03	08	AE	E8	00237	BLBS	STATUS, 17\$	
				009D	31	0023B	BRW	20\$	
		68	04	AE	3C	0023E	MOVZWL	SNUM, (HDR)	1177
		A8		01	B0	00242	MOVW	#1, 4(HDR)	1179
06	A8	20		02	A5	00246	MULW3	#2, 32(R10), 6(HDR)	1180
		08	0C	AE	B0	0024C	MOVW	START, 8(HDR)	1181
		01		02	90	00251	MOVB	#2, 1(ENT)	1185
		02		A9	8F	00255	MOVL	#-2132270760, 2(ENT)	1186
		06		A9	8F	0025D	MOVW	#9972, 6(ENT)	1187
				51	DD	00263	MOVL	ENT, R1	1188
				00000000G	EF	16	JSB	EXCH\$RT11 FORMAT CURRENT_DATE	
08	A9	56		08	A8	A3	SUBW3	8(HDR), BRUM, 8(ENT)	1189
		50		06	A8	3C	MOVZWL	6(HDR), R0	1190
		59		0E	A049	9E	MOVAB	14(R0)(ENT), ENT	
		51		01FE	C8	9E	MOVAB	510(R8), R1	1191
		51			59	D1	CMPL	ENT, R1	
					13	1F	BLSSU	18\$	
		7E		F7	8F	9A	MOVZBL	#247, -(SP)	
					01	DD	PUSHL	#1	
				00000000G	8F	DD	PUSHL	#EXCH\$ BADLOGIC	
	00000000G	00		03	FB	00291	CALLS	#3, LIB\$STOP	
	01	A9		08	90	00298	MOVB	#8, 1(ENT)	1192
		27		10	AE	E9	BLBC	16(SP), 19\$	1196
OC0E	C7	68		0200	8F	28	MOVW3	#512, (HDR), 3086(RTV)	1199
					6E	DD	PUSHL	(SP)	1200
	00000000G	EF			01	FB	CALLS	#1, EXCH\$RTACP_VERIFY_DIRECTORY	
		13			50	E8	BLBS	R0, 19\$	
		7E		F9	8F	9A	MOVZBL	#249, -(SP)	
					01	DD	PUSHL	#1	
				00000000G	8F	DD	PUSHL	#EXCH\$ BADLOGIC	
	00000000G	00		03	FB	002C0	CALLS	#3, LIB\$STOP	
				58	DD	002C7	PUSHL	HDR	1205
				01	DD	002C9	PUSHL	#1	
				06	DD	002CB	PUSHL	#6	
			OC	AE	DD	002CD	PUSHL	12(SP)	
	00000000G	EF		04	FB	002D0	CALLS	#4, EXCH\$IO_RT11_WRITE	
	08	AE		50	DD	002D7	MOVL	R0, STATUS	
		50		08	AE	DD	MOVL	STATUS, R0	1207
					04	002DF	RET		1208

; Routine Size: 736 bytes, Routine Base: EXCH\$INIT_CODE + 07B8

EXCHSINIT
V04-000

INIT verb dispatch and misc routines
init_rtl1_init

F 9
16-Sep-1984 00:59:01
14-Sep-1984 12:29:05

VAX-11 Bliss-32 V4.0-742
[EXCHNG.SRC]EXCINIT.B32;1

Page 40
(13)

EXC
V04

```
1126 1209 1 GLOBAL ROUTINE init_zero_home_blocks (start, buf) = %SBTTL 'init_zero_home_blocks (start, buf)'  
1127 1210 2 BEGIN  
1128 1211 3 ++  
1129 1212 4  
1130 1213 5 FUNCTIONAL DESCRIPTION:  
1131 1214 6  
1132 1215 7 Zero any possible Files-11 home blocks on the volume to prevent extraneous privilege problems with  
1133 1216 8 future mounts.  
1134 1217 9  
1135 1218 10 INPUTS:  
1136 1219 11  
1137 1220 12 start - the pbn of the first uninitialized block on the volume  
1138 1221 13 buf - the address of a 1024-byte buffer which has been set to zeroes  
1139 1222 14  
1140 1223 15 IMPLICIT INPUTS:  
1141 1224 16  
1142 1225 17 work area for INIT  
1143 1226 18  
1144 1227 19 OUTPUTS:  
1145 1228 20  
1146 1229 21 none  
1147 1230 22  
1148 1231 23 IMPLICIT OUTPUTS:  
1149 1232 24  
1150 1233 25 none  
1151 1234 26  
1152 1235 27 ROUTINE VALUE:  
1153 1236 28  
1154 1237 29 Success or worst error  
1155 1238 30  
1156 1239 31 SIDE EFFECTS:  
1157 1240 32  
1158 1241 33 disk blocks may be zeroed  
1159 1242 34 --  
1160 1243 35 $dbgtrc_prefix ('init_zero_home_blocks> ');  
1161 1244 36  
1162 1245 37 LOCAL  
1163 1246 38 blockfact, ! device blocking factor  
1164 1247 39 delta, ! home block search delta  
1165 1248 40 device_char : $bblock [dib$length], ! block for device characteristics  
1166 1249 41 devchar_desc : VECTOR [2, LONG], ! desc for above  
1167 1250 42 pbn, ! physical block number to check  
1168 1251 43 status  
1169 1252 44 ;  
1170 1253 45  
1171 1254 46 BIND  
1172 1255 47 init = exch$a_gbl [excg$a_init_work] : $ref_bblock, ! pointer to our work area  
1173 1256 48 volb = init [init$a_volb] : $ref_bblock ! pointer to exchange VOLB structure  
1174 1257 49 ;
```

```
1176 1258 2 ! For virtual volumes we cannot perform a normal home block scan, since the home block search sequence depen
1177 1259 2 ! the physical device geometry. This is unfortunate, since a virtual volume might be a copy of (and be copi
1178 1260 2 ! back to) a physical device. Usually, this copy will only be between a small disk (i.e. floppy or TU58) an
1179 1261 2 ! virtual volume. We will use our knowlege of these disks to perform ad hoc home block zeroing.
1180 1262 2
1181 1263 2 IF .volb [volb$v_virtual]
1182 1264 2 THEN
1183 1265 2 BEGIN
1184 1266 2     status = true; ! Assume success
1185 1267 2
1186 1268 2     SELECTONE .volb [volb$l_volmaxblock] OF
1187 1269 2     SET
1188 1270 2         [494] : IF .start LEQU 8 ! Single density floppy puts alternate home on pbn 8
1189 1271 2                 THEN
1190 1272 2                     status = exch$io_rtl1_write (.volb, 8, 1, .buf);
1191 1273 2
1192 1274 2         [988] : IF .start LEQU 15 ! Double density floppy puts alternate home on pbn 15
1193 1275 2                 THEN
1194 1276 2                     status = exch$io_rtl1_write (.volb, 15, 1, .buf);
1195 1277 2
1196 1278 2         [OTHERWISE] : ; ! Ignore large disks, TU58 puts home blocks on pbn 1 and 2 w
1197 1279 2                     ! we know that we have already hit
1198 1280 2     TES;
1199 1281 2
1200 1282 2     RETURN .status; ! All done with virtual volumes
1201 1283 2     END;
1202 1284 2
1203 1285 2 ! Read the device characteristics
1204 1286 2
1205 1287 2 devchar_desc [0] = dib$k_length; ! Init length of char buffer
1206 1288 2 devchar_desc [1] = device_char; ! and address of buffer
1207 1289 2
1208 1290 2 IF NOT (status = $getchn (chan=.volb [volb$w_channel], pribuf=devchar_desc))
1209 1291 2 THEN
1210 1292 2     $exch_signal_stop (.status);
```



```
1212 1293 2 (Home block geometry calculations borrowed from <INIT.SRC>RDHOME.B32)
1213 1294
1214 1295 Compute the home block search delta from the volume geometry in the device table. This is done according to
1215 1296 following rules, where volume geometry is expressed in the order sectors, tracks, cylinders:
1216 1297
1217 1298     n x 1 x 1:      1
1218 1299     1 x n x 1:      1
1219 1300     1 x 1 x n:      1
1220 1301
1221 1302     n x m x 1:      n+1
1222 1303     n x 1 x m:      n+1
1223 1304     1 x n x m:      n+1
1224 1305
1225 1306     s x t x c:      (t+1)*s+1
1226 1307
1227 1308 blockfact = (.device_char [dib$b_sectors]
1228 1309             * .device_char [dib$b_tracks]
1229 1310             * .device_char [dib$b_cylinders])
1230 1311             / .device_char [dib$l_maxblock];
1231 1312
1232 1313 delta = 1;
1233 1314 IF .device_char [dib$b_cylinders] GTR 1
1234 1315 AND
1235 1316     .device_char [dib$b_tracks] GTR 1
1236 1317 THEN
1237 1318     delta = .delta + .device_char [dib$b_tracks];
1238 1319
1239 1320 IF .device_char [dib$b_sectors] GTR 1
1240 1321 AND
1241 1322     (.device_char [dib$b_cylinders] GTR 1
1242 1323     OR
1243 1324     .device_char [dib$b_tracks] GTR 1)
1244 1325 THEN
1245 1326     delta = (.delta * .device_char [dib$b_sectors] + .blockfact) / .blockfact;
1246 1327
1247 1328 IF .delta EQL 0
1248 1329 OR
1249 1330     .delta GTRU .device_char [dib$l_maxblock] / 10
1250 1331 THEN
1251 1332     delta = 1;
1252 1333 $trace_print_fao ('block factor is !UL, delta is !UL', .blockfact, .delta);
1253 1334
1254 1335 ! Search for the home blocks to zero. To save time, we will just zap the first five possible positions for
1255 1336 home blocks. Note the potential hole: Disks with the home block far into the disk might not be completely
1256 1337 zeroed and might have protection anomalies. C'est la vie.
1257 1338
1258 1339 pbn = 1; ! Start search at pbn 1
1259 1340 DECR j FROM 4 TO 0
1260 1341 DO
1261 1342 BEGIN
1262 1343 $trace_print_fao ('index !UL, pbn !UL', .j, .pbn);
1263 1344 IF .start LEQU .pbn
1264 1345 THEN
1265 1346     IF NOT (status = exch$io_rtl1_write (.volb, .pbn, 1, .buf))
1266 1347     THEN
1267 1348         RETURN .status;
1268 1349 pbn = .pbn + .delta;
```

: 1269
: 1270
: 1271
: 12721350 2 END;
1351 2
1352 2 RETURN .status;
1353 1 END;

	56	00000000G	EF	9E	00000	
	5E	84	AE	9E	00002	
50	00000000G		10	C1	0000D	
50			04	C1	00015	
	53		60	D0	00019	
40	48		04	E1	0001C	
	51		01	D0	00021	
	50	44	A3	D0	00024	
	000001EE		50	D1	00028	
	8F		0F	12	0002F	
	08	04	AC	D1	00031	
		08	27	1A	00035	
			AC	DD	00037	
			01	DD	0003A	
			08	DD	0003C	
	000003DC	8F	16	11	0003E	
			50	D1	00040	
			15	12	00047	
		0F	04	AC	D1	00049
			0F	1A	0004D	
		08	AC	DD	0004F	
			01	DD	00052	
			0F	DD	00054	
			53	DD	00056	
	66		04	FB	00058	
	51		50	D0	0005B	
			00AC	31	0005E	
	6E	74	8F	9A	00061	
	04	AE	08	AE	9E	00065
			7E	7C	0006A	
		08	AE	9F	0006C	
			7E	D4	0006F	
		4A	A3	3C	00071	
	00000000G	00	05	FB	00075	
		51	50	D0	0007C	
		0A	51	E8	0007F	
			51	DD	00082	
	00000000G	00	01	FB	00084	
			04	0008B		
	50	10	AE	9A	0008C	
	52	11	AE	9A	00090	
	50		52	C4	00094	
	54	12	AE	3C	00097	
	50		54	C4	0009B	
54	50	78	AE	C7	0009E	
	52		01	D0	000A3	
			50	D4	000A6	

.EXTRN SYSSGETCHN, LIB\$STOP

.ENTRY	INIT_ZERO_HOME_BLOCKS, Save R2,R3,R4,R5,R6	1209
MOVAB	EXCH\$IO_RT11_WRITE, R6	
MOVAB	-124(SP), SP	
ADDL3	#16, EXCH\$A_GBL, R0	1255
ADDL3	#4, (R0), R0	1256
MOVL	(R0), R3	1263
BBC	#4, 72(R3), 4\$	
MOVL	#1, STATUS	1266
MOVL	68(R3), R0	1268
CMPL	R0, #494	1270
BNEQ	1\$	
CMPL	START, #8	
BGTRU	3\$	
PUSHL	BUF	1272
PUSHL	#1	
PUSHL	#8	
BRB	2\$	
CMPL	R0, #988	1274
BNEQ	3\$	
CMPL	START, #15	
BGTRU	3\$	
PUSHL	BUF	1276
PUSHL	#1	
PUSHL	#15	
PUSHL	R3	
CALLS	#4, EXCH\$IO_RT11_WRITE	
MOVL	R0, STATUS	
BRW	13\$	1282
MOVZBL	#116, DEVCHAR_DESC	1287
MOVAB	DEVICE_CHAR, DEVCHAR_DESC+4	1288
CLRQ	-(SP)	1290
PUSHAB	DEVCHAR_DESC	
CLRL	-(SP)	
MOVZWL	74(R3), -(SP)	
CALLS	#5, SYSSGETCHN	
MOVL	R0, STATUS	
BLBS	STATUS, 5\$	
PUSHL	STATUS	1292
CALLS	#1, LIB\$STOP	
RET		
MOVZBL	DEVICE_CHAR+8, R0	1309
MOVZBL	DEVICE_CHAR+9, R2	
MULL2	R2, R0	
MOVZWL	DEVICE_CHAR+10, R4	1310
MULL2	R4, R0	
DIVL3	DEVICE_CHAR+112, R0, BLOCKFACT	1311
MOVL	#1, DECTA	1313
CLRL	R0	1314

EXCH\$INIT
V04-000

INIT verb dispatch and misc routines
init_zero_home_blocks (start, buf)

K 9
16-Sep-1984 00:59:01
14-Sep-1984 12:29:05

VAX-11 Bliss-32 V4.0-742
[EXCHNG.SRC]EXCINIT.B32;1

Page 45
(16)

	01	12	AE	B1	000A8	CMPW	DEVICE_CHAR+10, #1	:	
			OF	1B	000AC	BLEQU	6\$:	
	01	11	50	D6	000AE	INCL	R0	:	1316
			AE	91	000B0	CMPB	DEVICE_CHAR+9, #1	:	
	55	11	07	1B	000B4	BLEQU	6\$:	1318
	52		AE	9A	000B6	MOVZBL	DEVICE_CHAR+9, R5	:	
	01	10	55	C0	000BA	ADDL2	R5, DELTA	:	1320
			AE	91	000BD	CMPB	DEVICE_CHAR+8, #1	:	
	06		17	1B	000C1	BLEQU	8\$:	1322
	01	11	50	E8	000C3	BLBS	R0, 7\$:	1324
			AE	91	000C6	CMPB	DEVICE_CHAR+9, #1	:	
	50	10	0E	1B	000CA	BLEQU	8\$:	1326
	50		AE	9A	000CC	MOVZBL	DEVICE_CHAR+8, R0	:	
	50		52	C4	000D0	MULL2	DELTA, R0	:	
52	50		54	C0	000D3	ADDL2	BLOCKFACT, R0	:	
			54	C7	000D6	DIVL3	BLOCKFACT, R0, DELTA	:	
			52	D5	000DA	TSTL	DELTA	:	1328
			0A	13	000DC	BEQL	9\$:	
50	78		AE	0A	C7	DIVL3	#10, DEVICE_CHAR+112, R0	:	1330
			50	52	D1	CMPL	DELTA, R0	:	
				03	1B	BLEQU	10\$:	
			52	01	D0	MOVL	#1, DELTA	:	1332
			54	01	D0	MOVL	#1, PBN	:	1339
			55	04	D0	MOVL	#4, J	:	1349
			54			CMPL	START, PBN	:	1344
		04	AC	D1	000F1	BGTRU	12\$:	
			10	1A	000F5	PUSHL	BUF	:	1346
		08	AC	DD	000F7	PUSHL	#1	:	
			01	DD	000FA	PUSHR	#^M<R3,R4>	:	
			18	BB	000FC	CALLS	#4, EXCH\$IO_RT11_WRITE	:	
	66		04	FB	000FE	MOVL	R0, STATUS	:	
	51		50	D0	00101	BLBC	STATUS, 13\$:	
	06		51	E9	00104	ADDL2	DELTA, PBN	:	1349
	54		52	C0	00107	SOBGEQ	J, 11\$:	1340
	E4		55	F4	0010A	MOVL	STATUS, R0	:	1352
	50		51	D0	0010D	RET		:	1353
			04	00	110			:	

; Routine Size: 273 bytes, Routine Base: EXCH\$INIT_CODE + 0A98

EXCH\$INIT
V04-000

INIT verb dispatch and misc routines
init_zero_home_blocks (start, buf)

L 9
16-Sep-1984 00:59:01
14-Sep-1984 12:29:05

VAX-11 Bliss-32 V4.0-742
[EXCHNG.SRC]EXCINIT.B32;1

Page 46
(17)

: 1274 1354 1 END
: 1275 1355 0 ELUDOM

.EXTRN LIB\$SIGNAL, LIB\$STOP

PSECT SUMMARY

Name	Bytes	Attributes
EXCH\$INIT_PLIT	286 NOVEC,NOWRT, RD ;	EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)
EXCH\$INIT_CODE	2985 NOVEC,NOWRT, RD ;	EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	122	0	1000	00:01.8
\$255\$DUA28:[EXCHNG.OBJ]EXCLIB.L32;1	1151	142	12	79	00:01.3

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LISS:EXCINIT/OBJ=OBJ\$:EXCINIT MSRC\$:EXCINIT/UPDATE=(ENHS:EXCINIT)

: Size: 2985 code + 286 data bytes
: Run Time: 00:55.5
: Elapsed Time: 03:18.7
: Lines/CPU Min: 1465
: Lexemes/CPU-Min: 25197
: Memory Used: 279 pages
: Compilation Complete

0161 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY